(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GE/ENG/88D-49

ANALYSIS AND SIMULATION OF AN AUDIO ADAPTIVE EQUALIZER

THESIS

John R. Strasburger
Captain, USAF

AFIT/GE/ENG/88D-49

Approved for public release; distribution unlimited

AFIT/GE/ENG/88D-49

ANALYSIS AND SIMULATION OF AN AUDIO ADAPTIVE EQUALIZER

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

John R. Strasburger
Captain, USAF

December 1988

Acknowledgements

This research would not have been possible without the assistance , advice and support of many people. In particular, I would like to thank my thesis advisor, Captain Robert Williams for his assistance and suggestions which were paramount for the thesis completion. Also Richard Mckinley and Joe Steuvers of the Armstrong Aerospace Medical Research Laboratory provided guidance and support at key stages of my effort. Most of all I want to thank my wife Madonna for her patience and understanding during this trying period.

John R. Strasburger

Table of Contents

Page

List of Figures

Figure Page

Figure Page

List of Tables

AFIT/GE/ENG/88D-49

Abstract

The purpose of this thesis was to explore the feasibility of replacing a manual audio equalizer with an adaptive inverse filter that adaptively equalizes the spectral distortion of an audio system. The impulse response of an audio system , which includes the response of the speaker crossover network, the power amplifiers, speakers, and the acoustic transfer function between the system's speakers and a reference microphone, distorts an audio system's input signal spectrum. The Adaptive Inverse Prefilter, the Filtered-x algorithm, and the Adaptive Inverse Modeling Control System are investigated which remove the distortion by pre-filtering the audio system's input signal with the audio system's inverse. The audio system examined is the Armstrong Aerospace Medical Research Laboratory's Performance and Communication Research and Technology reverberation chamber facility located at Wright Patterson Air Force Base.

The researcher presents two innovative solutions: a multi-band Adaptive Inverse Modeling Control System (AIMCS) and a frequency domain adaptive spectrum shaper. The adaptive spectrum shaper uses an improved weight update algorithm developed specifically for this application. Computer simulation results are presented which demonstrate the effectiveness of the multi-band AIMCS and the adaptive spectrum shaper in removing the spectral distortion of an audio system model.

ANALYSIS AND SIMULATION OF AN
AUDIO ADAPTIVE EQUALIZER

I. Introduction

Background

The Biological Acoustics Branch of the Armstrong Aerospace Medical Research Laboratory, AAMRL/BBA, tests the intelligibility performance of aircraft audio and radio systems in a simulated, cockpit, ambient noise environment. A trained panel of ten subjects comprised of nine listeners and one talker, who are located in a large reverberation chamber, evaluate the intelligibility effectiveness of aircraft communication links which include aircrew microphones and headsets, aircraft intercommunication sets and radios. The intelligibility testing is conducted in a pink noise or a simulated aircraft noise environment. For some simulated aircraft noise environments, actual cockpit recordings are played through the reverberation chamber's audio system.

The major components of the noise generation system excluding the tape deck for the cockpit recording playback and the speaker crossover network are shown in Figure 1. A complete block diagram of the audio system is presented in Chapter III. The analog white noise generator produces white Gaussian noise over the 20 to 50,000 Hz frequency range. The white noise is filtered through a pink noise filter which rolls off at 3 db per octave from 20 Hz to 20 KHz. The

pink noise is then spectral shaped with a 32 band graphic equalizer. The spectral shaped noise is amplified with high power amplifiers to provide up to 125 dB sound pressure level in the reverberation chamber. The audio system components from point A to point C in Figure 1 including the acoustic transfer function between the speakers and the microphone is referred throughout the thesis as the audio plant.

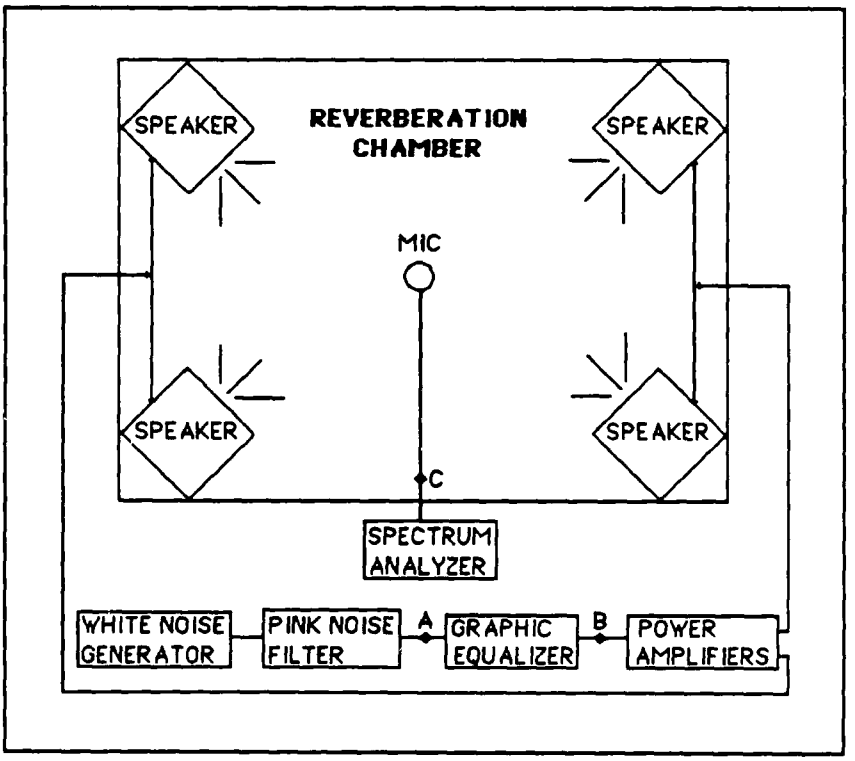


Figure 1. Noise Generation System

The spectral shape of the reverberation chamber's noise spectrum is monitored with a spectrum analyzer connected to a reference microphone located in the center of the chamber. The

graphic equalizer is adjusted to achieve the desired spectral shape of the pink noise spectrum at the microphone output.

Intelligibility testing must be carefully monitored to insure the validity and the repeatability of the tests. One test control parameter, that is crucial for valid test results, is the equalization of the reverberation chamber's noise spectrum using the graphic equalizer. The current equalization procedure is time consuming and complicated. It requires readjustment for each intelligibility test run to compensate for changes in the chamber's temperature and humidity and for equipment drift. A preliminary BBA investigation suggests that the equalization could be accomplished automatically with an adaptive filter (7).

Purpose

The purpose of this thesis is to explore the feasibility of using an adaptive pre-inverse filter to remove the frequency distortion effects of the audio plant's transfer function so that the plant's output at the reference microphone has the same spectral shape as the input noise spectrum.

Scope

The scope of this thesis is limited to an investigation of finite impulse response (FIR) least mean square (LMS) adaptive filters. Applicable LMS adaptive filters are simulated and tested using the FORTRAN programming language and digitized pink noise data from the reverberation chamber. The maximum number of taps for the

adaptive filter design is limited to the maximum number practical with a dedicated digital signal processor chip. No hardware is produced. The results are computer models, listings, and plots.

Assumptions

For the thesis research, the following assumptions have been made:

1. The impulse response of the audio system varies slowly.
2. The audio system's impulse response and its inverse can be characterized with a finite impulse response filter .
3. Effects of finite-word length, round off, and quantization can be ignored.

Approach

The thesis approach consists of a review of current literature to identify promising adaptive LMS filter candidates, the development of adaptive filter theory, and the simulation and analyzes of the selected adaptive inverse filter configurations.

The literature review identified the Adaptive Modeling Inverse Control System (AIMCS) and the filtered-x algorithm as the most promising adaptive filter solutions for the simulation and analysis. Both filter configurations have the required input output structure in which the inverse adaptive filter is in front and in series with the unknown plant. In addition to the AIMCS and the filtered-x algorithm, the Adaptive Inverse Pre-filter (AIP) is also explored as a

potential inverse filter solution. The findings of the literature review are integrated with the theory developed in Chapter II.

The theoretical development in Chapter II provides the foundation required to analyze the AIMCS, filtered-x algorithm, and the AIP. The initial background theory introduces the adaptive linear combiner and the LMS algorithm. The theory then focuses on the AIP, filtered-x algorithm, and the AIMCS.

An adaptive system is justified if the audio plant's response distorts the input signal and changes with time. To confirm that an adaptive system is warranted, the spectral response of the audio plant is characterized by analyzing digitized data from the audio plant. The digitized data is also used to generate the FIR plant models and in the adaptive inverse filter simulations.

Computer simulations are conducted to test the effectiveness of the three adaptive inverse filters configurations in removing the distortion of the plant. The merit of each adaptive inverse filter simulation is assessed by comparing the desired spectrum with the spectrum at the output of the plant.

The simulation programs are written in FORTRAN 77 and are compiled and executed on the Elxsi computer. Modular top down programming techniques and descriptive comments enhance the readability of the source code. The simulation programs are composed of a main program in which the variables are declared, the input output data files are opened, the data reads and writes are performed and the adaptive filter simulations are implemented. Within the main program loop, calls are made to signal analysis, and noise generation subroutines.

The thesis concludes by recommending a course of action for continued research. For this specific application, it appears that a frequency domain implementation has significant advantages over the time domain approaches described within this thesis.

II. Theory

This chapter discusses the adaptive linear combiner, the LMS algorithm, and adaptive inverse filter theory. The adaptive linear combiner and the LMS algorithm are first briefly introduced. The remainder of the chapter then focuses on a development of adaptive inverse filter theory which is applicable to this thesis.

Adaptive Linear Combiner

The adaptive linear combiner is the basic building block for adaptive signal processing systems. The adaptive linear combiner consists of tap delay line with adjustable weights and a summing unit (14:16-19). A single input adaptive linear combiner with a desired response signal d_k and an error signal e_k is illustrated in Figure 2. The adaptive linear combiner weights are adapted to

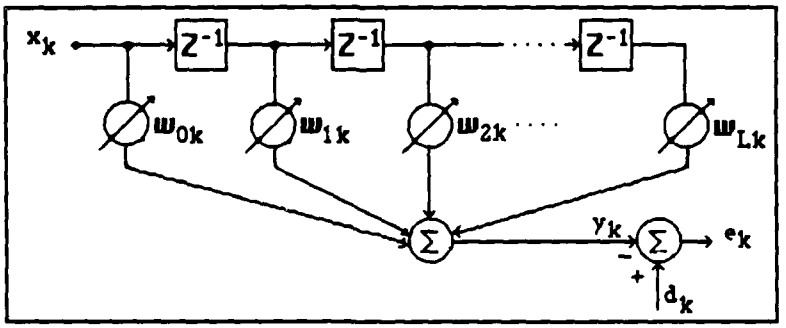


Figure 2. Adaptive Linear Combiner (14:17)

minimize the error e_k between the desired response signal d_k and the output of the adaptive linear combiner y_k.

The kth output signal y_k is given by

$$y_k = \sum_{n=0}^{L} w_{nk} x_{k-n} = \mathbf{W}_k^T \mathbf{X}_k = \mathbf{X}_k^T \mathbf{W}_k \qquad (2.1)$$

where $$\mathbf{X}_k = [\, x_k \quad x_{k-1} \quad \ldots \quad x_{k-L} \,]^T \qquad (2.2)$$

and $$\mathbf{W}_k = [\, w_{0k} \quad w_{1k} \quad \ldots \quad w_{Lk} \,]^T \qquad (2.3)$$

In Eqs(2.1-2.3), the superscript ^{T} denotes the transpose matrix operator. The error signal at the kth time is

$$e_k = d_k - y_k \qquad (2.4)$$

Substitution of Eq(2.1) into Eq(2.4) yields

$$e_k = d_k - \mathbf{W}_k^T \mathbf{X}_k \qquad (2.5)$$

For the rest of this development, the subscript k will be removed from the weight notation since it has been assumed that the weights have converged and are no longer adapting. The instantaneous squared error is found by squaring Eq (2.5):

$$e_k^2 = d_k^2 + \mathbf{W}^T \mathbf{X}_k \mathbf{X}_k^T \mathbf{W} - 2 d_k \mathbf{X}_k^T \mathbf{W} \qquad (2.6)$$

Assuming $\mathbf{X}_k$ and $\mathbf{W}$ are uncorrelated and $\mathbf{X}_k$ is zero mean and stationary, the mean squared error (MSE) is derived by taking the the expected value of Eq (2.6) (14:20):

$$\text{MSE} = \xi = E[e_k^2] = E[d_k^2] + \mathbf{W}^T E[\mathbf{X}_k \mathbf{X}_k^T]\mathbf{W} - 2E[d_k \mathbf{X}_k^T]\mathbf{W}$$

$$\xi = E[e_k^2] = E[d_k^2] - \mathbf{W}^T \mathbf{R}\mathbf{W} - 2\mathbf{P}^T\mathbf{W} \qquad (2.7)$$

where the input autocorrelation matrix $\mathbf{R}$ is given by

$$E[\mathbf{X}_k \mathbf{X}_k^T] = \mathbf{R} = E\begin{bmatrix} x^2_k & x_k x_{k-1} & \cdot & \cdot & \cdot & x_k x_{k-L} \\ x_{k-1} x_k & x^2_{k-1} & \cdot & \cdot & \cdot & x_{k-1} x_{k-L} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{k-L} x_k & x_{k-L} x_{k-1} & \cdot & \cdot & \cdot & x^2_{k-L} \end{bmatrix} \qquad (2.8)$$

and the cross correlation vector $\mathbf{P}$ between the input and the desired signal is given by

$$E[d_k \mathbf{X}_k^T] = \mathbf{P} = E\begin{bmatrix} d_k x_k \\ d_k x_{k-1} \\ \cdot \\ \cdot \\ d_k x_{k-L} \end{bmatrix} \qquad (2.9)$$

Equation (2.7) shows that the mean-square error is a quadratic function of the components of the weight vectors (14:20). The quadratic function has a paraboloid performance surface for a two weight adaptive linear combiner and hyperparaboloid performance surface for an adaptive linear combiner with three or more weights. Figure 3 illustrates a typical performance surface for a two weight adaptive linear combiner in which the vertical axis represents the

mean-square error and the two horizontal axes are the weight values. The performance surface contour is bowl shaped, with the concave upward, and only one minimum at the bottom of the bowl. The optimum weights $\mathbf{W}^*$ can be determined by projecting the minimum MSE on the weight vector plane. In the next section, the least mean square (LMS) algorithm that seeks the minimum of the performance surface is discussed.

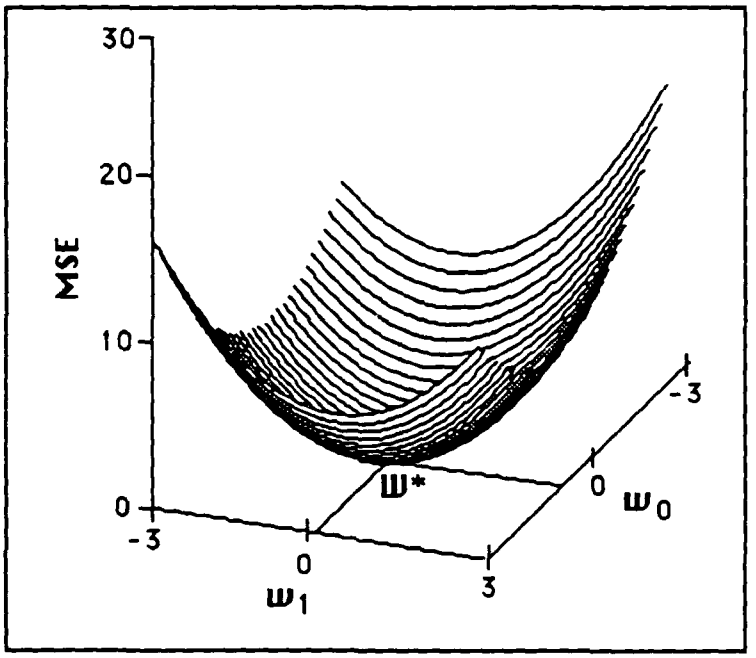


Figure 3. Portion of a Two Dimensional Quadratic Performance Surface

Least Mean Square Algorithm

The LMS algorithm is an approximation of the steepest descent iterative algorithm which searches the performance surface for the weight vector that minimizes the mean square error between the

desired response signal d_k and the adaptive filter output signal y_k (14:99). The LMS algorithm utilizes gradient estimates to descend down the performance surface and locate the minimum. Bernard Widrow and Samuel Streans describe the LMS algorithm as an elegantly simple search method for adaptive signal processing applications, where the adaptive system is an adaptive linear combiner with both the input state vector $\mathbf{x}_k$ and the desired signal d_k available at each iteration (14:99).

The steep descent algorithms change the weight vector in proportion to the negative gradient vector

$$\mathbf{W}_{k+1} = \mathbf{W}_k - u\nabla_k \tag{2.10}$$

where $\mathbf{W}_{k+1}$ is the estimated weight vector for the k+1th iteration, $\mathbf{W}_k$ is the weight vector at the kth iteration, u is the gain constant which regulates the rate and stability of convergence, and ∇_k is the gradient vector at w_k (14:48). The gain constant u has units of reciprocal power.

The LMS algorithm uses e^2_k as an estimate of ξ to calculate the gradient at each iteration. The gradient is obtained by taking the derivative of Eq(2.5) with respect to the weight vector (14:100)

$$\nabla_k = \begin{bmatrix} \frac{\partial e^2_k}{\partial w_0} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial e^2_k}{\partial w_L} \end{bmatrix} = 2e_k \begin{bmatrix} \frac{\partial e_k}{\partial w_0} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial e_k}{\partial w_L} \end{bmatrix} = -2e_k\mathbf{X}_k \tag{2.11}$$

Substitution of the gradient Eq(2.11) into Eq(2.10) yields the LMS algorithm

$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2ue_k\mathbf{X}_k \qquad (2.12)$$

Because the LMS algorithm's gradient is an estimate, the adaptive process is noisy and does not follow the true path of steepest descent (14:100). However, the LMS algorithm is practically realized without complex mathematical computations. The upcoming sections apply the basic adaptive signal processing concepts just introduced in the development of adaptive inverse filter theory.

Inverse Filter Theory

The LMS algorithm is utilized in a multitude of signal processing applications which includes interference and echo cancellation, modeling and inverse filtering of an unknown propagation channel, linear prediction and spectral estimation (14;4:1281-1283). This section focuses on adaptive inverse filters.

An inverse filter can remove the undesired distortion effects of a plant. The distortion for this thesis is the non-flat frequency response of the audio plant. The audio plant consists of the audio amplifiers, speaker crossover network, speakers, and the acoustic transfer function between the speakers and the reference microphone in the reverberation chamber. To solve this thesis problem, the adaptive inverse filter must be placed in front of the

audio plant to pre-equalize the audio plant's input audio signal; so, the output of the audio plant at the microphone has the desired spectral shape.

An adaptive inverse filter is required when the plant characteristics are unknown and/or change slowly with time. The transfer function of the audio equipment and the reverberation chamber acoustics changes with time as a result of air temperature and humidity variations, audio equipment drift, and number and location of the subjects in the reverberation chamber (7). In Chapter 4, actual test data from the reverberation chamber verify that the plant's transfer function does vary with time.

General Inverse Filter Theory. Figure 4 illustrates the inverse filter concept. An inverse filter W(z) is placed in series with the

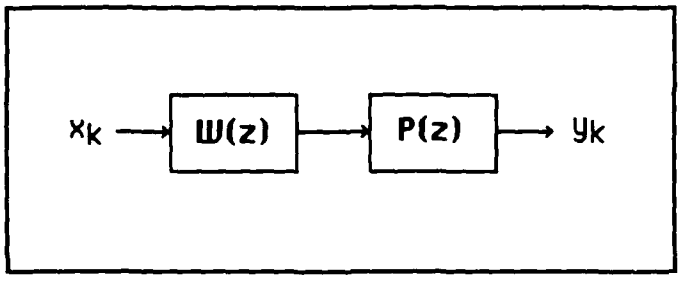


Figure 4. Inverse Filter

plant P(z) to remove the effect of the plant; so, the output signal y_k is equal to the input signal x_k (5:422-448). For the inverse filter to exactly compensate for the plant, the product of the z transforms W(z) and P(z) must be equal to unity

$$W(z)\,P(z) = P(z)\,W(z) = 1 \qquad (2.13)$$

Thus, the series connection of the inverse filter and the plant act as a straight wire. Solving Eq(2.13) for the inverse filter transfer function gives

$$W(z) = \frac{1}{P(z)} = P(z)^{-1} \tag{2.14}$$

Hence, the inverse filter's transfer function is the reciprocal of the plant's transfer function. For illustrative purposes, let the plant's transfer function be given by the ratio of two polynomial expressions in z^{-1}

$$P(z) = \frac{D(z)}{F(z)} \tag{2.15}$$

To exactly cancel the plant, the inverse filter's transfer function is given by

$$W(z) = \frac{F(z)}{D(z)} \tag{2.16}$$

Thus, the poles of the inverse filter transfer function are the zeros of the plant and the zeros of the inverse filter are the poles of the plant.

Minimum and Non-minimum Phase Plants. The phase response of the plant has important ramifications on the design of inverse filters. If the plant zeros are contained within the unit circle, the plant has minimum phase and the corresponding inverse filter's poles are contained within the unit circle (5:426-429). Because the

poles are contained within the unit circle, the minimum phase plant's inverse is stable. For a given plant's magnitude response, a minimum phase plant is a causal system with the smallest phase response possible (5:427). A non-minimum phase plant has a zero or zeros outside the unit circle. Therefore, the inverse filter for a non-minimum phase plant has poles outside of the unit circle to cancel the zeros which are outside the unit circle. If the inverse is not exact, it will be unstable. An exact inverse can not be practically realized. A method to design an approximate stable inverse for the non-minimum phase plant is discussed in the next paragraph. A minimum phase plant's phase response is characterized as a continuous function of the radian frequency ω which starts at zero phase at $\omega = 0$, returns to zero phase at $\omega = \pi$, and does not exceed π radians (5:426). While a non-minimum phase response is characterized as a continuous function of the radian frequency (ω) which starts at zero phase at $\omega = 0$ and ends at $-N\pi$ phase at $\omega = \pi$ where N is the number of zeros outside of the unit circle (5:427). Zeros on the unit circle result in discontinuous jumps of π radians in the phase response.

The inverse z transform for the inverse filter of a non-minimum plant yields an infinite left sided or two sided non-causal impulse response (14:233,238). The impulse response is left sided when all the zeros are outside the unit circle and two sided when the zeros are located inside and outside of the unit circle. By delaying the inverse impulse response to shift the impulse response to the right and truncating the infinite impulse response, an approximate, delayed, causal inverse can be realized with an FIR filter (14:233).

Stephen Nealy and Jont Allen describe the impulse response of a room as having non-minimum phase when the microphone is more than 8 in. from the speakers (9:169). In the reverberation chamber at AFAMRL/BBA, the microphone is separated from the speakers by more than 6 ft. Therefore, the impulse response of the reverberation chamber has non-minimum phase characteristics which must be considered in designing the inverse filter. The next section analyzes three adaptive inverse filter configurations and addresses the limitations of the Adaptive Inverse Prefilter.

Adaptive Inverse Pre-filter. The adaptive inverse pre-filter (AIP) is illustrated in Figure 5 where a plant occurs after and in series with the adaptive inverse filter. Bernard Widrow and Samuel Stearns state that AIP is "almost guaranteed to be unstable or, if not, to converge to an irrelevant solution" (14:289). Because of some preliminary simulations in which the AIP converged to a relevant solution, the AIP configuration is analyzed to determine its limitations.

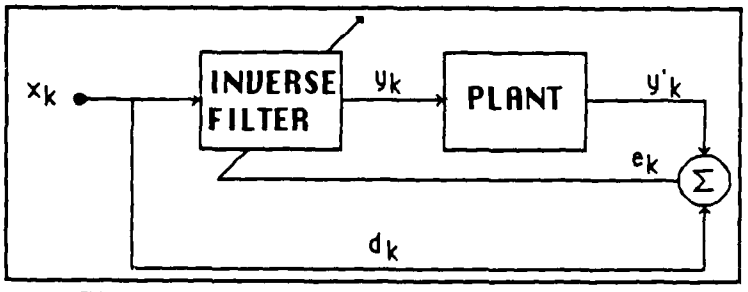


Figure 5. Adaptive Inverse Pre-filter

Derivation of the AIP Optimum Weight Transfer Function. The purpose of the derivation is to determine whether the AIP inverse

filter can inverse filter an FIR plant. The derivation follows closely the derivation of an optimum weight transfer function for the adaptive inverse modeling filter in the text by Widrow and Stearns (14:234-235).

The plant P(z) has weights p_n where n denotes the nth weight. From Figure 5, the error signal e_k is given by

$$e_k = d_k - y'_k \tag{2.17}$$

where

$$y'_k = \sum_{n=0}^{L} p_n \; y_{(k-n)} \tag{2.18}$$

and

$$y_k = \sum_{m=0}^{J} w_{m(k)} \; x_{(k-m)} \tag{2.19}$$

Substitution of Eqs (2.18) and (2.19) into Eq (2.17) yields

$$e_k = d_k - \sum_{n=0}^{L} \sum_{m=0}^{J} p_n \; w_{m(k-n)} \; x_{(k-m-n)} \tag{2.20}$$

Squaring Eq(2.20) and substituting x_k for d_k gives the instantaneous squared error

$$e^2_k = x^2_k - 2x_k \sum_{n=0}^{L} \sum_{m=0}^{J} p_n \; w_{m(k-n)} \; x_{(k-m-n)}$$

$$+ \sum_{n=0}^{L} \sum_{m=0}^{J} \sum_{l=0}^{L} \sum_{p=0}^{J} p_n p_l w_{m(k-n)} w_{p(k-l)} x_{(k-m-n)} x_{(k-l-p)} \tag{2.21}$$

Assuming the filter has converged to the optimum weights so the weights are no longer time dependent and taking the expected value of Eq(2.21) yields

$$\xi = E(e^2_k) = E(x^2_k) - 2 \sum_{n=0}^{L} \sum_{m=0}^{J} p_n \; w_m \; E(x_k \; x_{k-m-n})$$

$$+ \sum_{n=0}^{L} \sum_{m=0}^{J} \sum_{l=0}^{L} \sum_{p=0}^{J} p_n \; p_l \; w_m \; w_p \; E(x_{k-m-n} \; x_{k-l-p}) \quad (2.22)$$

Substitution of the correlation function for the expected value function (i.e. $\phi_{xx}(n) = E[x_k \; x_{k+n}]$) gives (14:128)

$$\xi = \phi_{xx}(0) - 2 \sum_{n=0}^{L} \sum_{m=0}^{J} p_n \; w_m \; \phi_{xx(-m-n)}$$

$$+ \sum_{n=0}^{L} \sum_{m=0}^{J} \sum_{l=0}^{L} \sum_{p=0}^{J} p_n \; p_l \; w_m \; w_p \; \phi_{xx(-m-n+l+p)} \quad (2.23)$$

The least mean square ideal weight vector W^* is obtained by substituting k for m, by applying the symmetry relationship for the autocorrelation function, $\phi_{xx(-k-n+l+p)} = \phi_{xx(k+n-l-p)}$, by taking the gradient of the least mean square error performance surface, and by setting the gradient equal to zero (14:234)

$$\frac{\partial \xi}{\partial w_k} = -2 \sum_{n=0}^{L} p_n \; \phi_{xx(-k-n)}$$

$$+ 2 \sum_{n=0}^{L} \sum_{l=0}^{L} \sum_{p=0}^{J} p_n \; p_l \; w_p \; \phi_{xx(k+n-l-p)} = 0 \quad (2.24)$$

which simplifies to

$$\sum_{n=0}^{L} p_n \, \phi_{xx(-k-n)} = \sum_{n=0}^{L} \sum_{l=0}^{L} \sum_{p=0}^{J} p_n \, p_l \, w_p^* \, \phi_{xx(k+n-l-p)} \qquad (2.25)$$

To express Eq(2.25) in terms of z transforms, Equation's (2.25) finite limits of summation are replaced with infinite limits by making the plant and adaptive filter coefficients zero outside the finite limit boundaries (14:120)

$$[p_n] = [\ldots 0\,0\,0\,p_0\,p_1 \ldots p_L\,0\,0\,0\ldots] \qquad (2.26)$$

$$[p_l] = [\ldots 0\,0\,0\,p_0\,p_1 \ldots p_L\,0\,0\,0\ldots]$$

$$[w_p^*] = [\ldots 0\,0\,0\,w_0\,w_1 \ldots w_J\,0\,0\,0\ldots]$$

Thus, Eq(2.25) becomes

$$\sum_{n=-\infty}^{\infty} p_n \, \phi_{xx(-k-n)} = \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \sum_{p=-\infty}^{\infty} p_n \, p_l \, w_p^* \, \phi_{xx(k+n-l-p)} \qquad (2.27)$$

Taking the z transform of both sides gives

$$\sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} p_n \phi_{xx(-k-n)} \, z^{-k} = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \times \sum_{p=-\infty}^{\infty} p_n p_l w_p^* \phi_{xx(k+n-l-p)} \, z^{-k} \qquad (2.28)$$

To simplify the left side of Eq (2.28), let $m = k + n$ to obtain

$$\sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} p_n \phi_{xx(-m)}\, z^{n-m} = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \times \sum_{p=-\infty}^{\infty} p_n p_l w_p^* \phi_{xx(k+n-l-p)} z^{-k} \quad (2.29)$$

Applying the symmetry relationship for the autocorrelation function yields

$$\sum_{n=-\infty}^{\infty} p_n z^n \sum_{m=-\infty}^{\infty} \phi_{xx(m)}\, z^{-m} = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \times \sum_{p=-\infty}^{\infty} p_n p_l w_p^* \phi_{xx(k+n-l-p)}\, z^{-k} \quad (2.30)$$

, and applying the z transform gives

$$P(z^{-1})\Phi_{xx}(z) = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \sum_{p=-\infty}^{\infty} p_n p_l w_p^* \phi_{xx(k+n-l-p)}\, z^{-k} \quad (2.31)$$

To simplify the right side of Eq (2.31), let $r = k + n - l - p$.

$$P(z^{-1})\Phi_{xx}(z) = \sum_{r=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \sum_{p=-\infty}^{\infty} p_n p_l w_p^* \phi_{xx(r)}\, z^{-r-l-p+n} \quad (2.32)$$

Collecting terms

$$P(z^{-1})\Phi_{xx}(z) = \sum_{n=-\infty}^{\infty} p_n z^n \sum_{l=-\infty}^{\infty} p_l z^{-l} \sum_{p=-\infty}^{\infty} w_p^* z^{-p} \sum_{r=-\infty}^{\infty} \phi_{xx(r)} z^{-r} \quad (2.33)$$

, applying the z transform

$$P(z^{-1})\Phi_{xx}(z) = P(z^{-1})P(z)W^*(z)\Phi_{xx}(z) \qquad (2.34)$$

, and simplifying gives the expected optimal transfer function which is the reciprocal of the plant transfer function

$$W^*(z) = \frac{1}{P(z)} \qquad (2.35)$$

Since the impulse response length of the adaptive filter is finite, $W^*(z)$, which is implemented with an FIR filter, can only approximate the infinite impulse response of the plant's inverse, the mean squared error ξ_k will not be zero but will approach the minimum on the average after the adaptive filter weights have converged to the optimum solution. An equation for the performance function for the finite length FIR adaptive inverse filter is derived in the next section. The performance function equation expresses the mean squared error as quadratic function of the linear combiner weights.

Performance Function Equation Derivation. From Figure 5, the error signal is given by Equation (2.17) which is restated for convenience

$$e_k = d_k - y'_k \qquad (2.17)$$

The plant output y'_k is given by

$$y'_k = x_k * w_k * p_k \qquad (2.36)$$

Using the convolution commutative law, $f(t) * g(t) = g(t) * f(t)$, and the associative law, $f(t) * [g(t) * h(t)] = [f(t) * g(t)] * h(t)$, Eq(2.36) becomes (11:105)

$$y'_k = p_k * x_k * w_k \tag{2.37}$$

which assumes the plant and the adaptive filter can be commuted. Equation(2.37) in matrix form is given by

$$y'_k = \mathbf{P}^T \chi_k \mathbf{W}_k \tag{2.38}$$

where $\mathbf{P}$ is the plant weight vector

$$\mathbf{P}_k = \begin{bmatrix} p_{0k} \\ p_{1k} \\ . \\ . \\ . \\ p_{Lk} \end{bmatrix} \tag{2.39}$$

and χ_k is input matrix

$$\chi_k = \begin{bmatrix} x_k & x_{k-1} & . & . & . & x_{k-L} \\ x_{k-1} & x_{k-2} & . & . & . & x_{k-L-1} \\ . & . & . & . & . & . \\ x_{k-L} & x_{k-L-1} & . & . & . & x_{k-2L} \end{bmatrix} \tag{2.40}$$

where $\chi_k = \chi_k^T$. Substitution of Eq(2.38) into Eq(2.17) yields

$$e_k = d_k - \mathbf{P}^T \chi_k \mathbf{W} \tag{2.41}$$

Equation (2.41) is squared to obtain the instantaneous squared error

$$e_k^2 = d_k^2 + \mathbf{W}^T\chi_k\mathbf{P}\mathbf{P}^T\chi_k^T\mathbf{W} - 2d_k\mathbf{P}^T\chi_k\mathbf{W} \qquad (2.42)$$

Assuming e_k , d_k, and χ_k are wide sense stationary the expected value of Eq(2.42) is given by

$$\xi = E[e_k^2] = E[d_k^2] + \mathbf{W}^TE[\chi_k\mathbf{P}\mathbf{P}^T\chi_k^T]\mathbf{W} - 2E[d_k\mathbf{P}^T\chi_k]\mathbf{W}_k \qquad (2.43)$$

Defining **R** as the filtered input correlation matrix

$$\mathbf{R} = E[\chi_k\mathbf{P}\mathbf{P}^T\chi_k^T] \qquad (2.44)$$

and **K** as the filtered cross correlation vector

$$\mathbf{K} = E[d_k\mathbf{P}^T\chi_k]^T \qquad (2.45)$$

Eq(2.43) reduces to the performance function

$$\xi = E[e_k^2] = E[d_k^2] + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{K}^T\mathbf{W} \qquad (2.46)$$

To find the minimum, the gradient of Eq(2.46), $\frac{\partial \xi}{\partial \mathbf{W}}$, is set equal to zero

$$0 = 2\mathbf{W}^T\mathbf{R} - 2\mathbf{K}^T \tag{2.47}$$

to obtain the Weiner weight vector $\mathbf{W}^*$

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{K}^T \tag{2.48}$$

where it was assumed that $\mathbf{R}$ is invertible. Substitution of Eq(2.48) into Eq(2.46) yields an expression for the minimum mean squared error ξ_{min} (14:22)

$$\xi_{min} = E[d_k^2] - \mathbf{K}^T\mathbf{W}^* \tag{2.49}$$

Convergence of the LMS Weight Update Equation. In the following development, it will be shown that convergence of the AIP configuration to the optimum weight vector Eq (2.48) is not possible with a non-minimum phase plant or a plant with a transport delay when the input signal is white, zero mean noise. The non-convergence is the result of the plant's phase response which decorrelates the X_k and y'_k weight update LMS inputs.

To illustrate the decorrelative effect of the non-minimum phase plant, consider the following example where a two tap adaptive filter W(z) is attempting to inverse filter a two tap FIR plant. The filter configuration is shown in Figure 6.

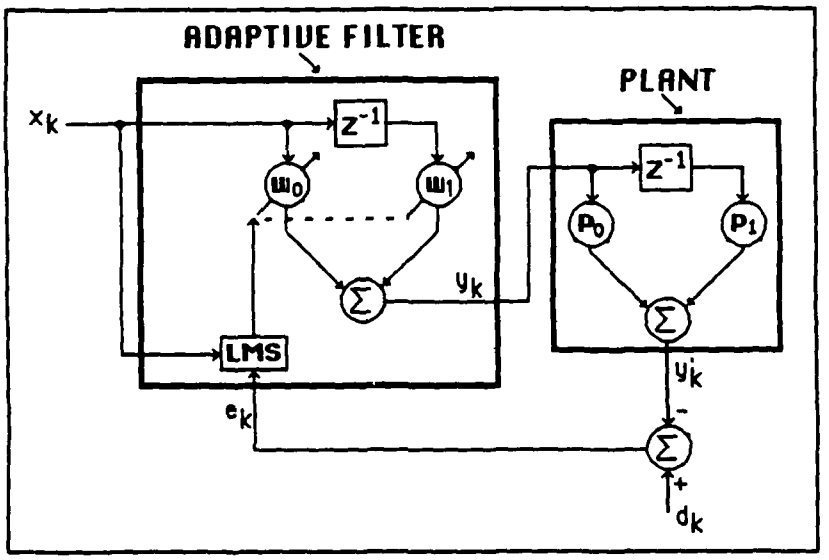


Figure 6. Two Tap AIP Inverse Filtering a Two Tap Plant

From Figure 6, the plant weight vector is $P = [\, p0 \;\; p1 \,]^T$. Substitution of the plant weight vector into Eq(2.41) yields

$$e_k = d_k - \begin{bmatrix} p0 \\ p1 \end{bmatrix}^T \begin{bmatrix} x_k & x_{k-1} \\ x_{k-1} & x_{k-2} \end{bmatrix} \begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} \qquad (2.50)$$

The weight update equation for the AIP configuration is obtained by substituting Eq(2.50) into Eq(2.12)

$$\begin{bmatrix} w_{0k+1} \\ w_{1k+1} \end{bmatrix} = \begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} + 2u \left(d_k \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} - \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}^T \begin{bmatrix} x_k & x_{k-1} \\ x_{k-1} & x_{k-2} \end{bmatrix} \begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} \right) \quad (2.51)$$

The expected value of Eq(2.51) is given by

$$E\begin{bmatrix} w_{0k+1} \\ w_{1k+1} \end{bmatrix} = E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} + 2uE\left(d_k \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} - \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}^T \begin{bmatrix} x_k & x_{k-1} \\ x_{k-1} & x_{k-2} \end{bmatrix} \begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} \right) \quad (2.52)$$

Reexpressing Eq(2.12) as

$$\mathbf{W}_k = \mathbf{W}_0 + 2u \sum_{j=0}^{k-1} e_j \mathbf{X}_j \quad (2.53)$$

it is shown that $\mathbf{W}_k$ is dependent on $\mathbf{X}_{k-1}, \mathbf{X}_{k-2} \ldots, \mathbf{X}_0$ and not on $\mathbf{X}_k$ (15:187). Since $\mathbf{W}_k$ and $\mathbf{X}_k$ are statistically independent, the expected value of the product is equal to the product of the expected values (14:20). Thus, Eq(2.52) becomes

$$E\begin{bmatrix} w_{0k+1} \\ w_{1k+1} \end{bmatrix} = E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} + 2u\,E\begin{bmatrix} d_k x_k \\ d_k x_{k-1} \end{bmatrix} - 2u\,E\left(\begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}^T \begin{bmatrix} x_k & x_{k-1} \\ x_{k-1} & x_{k-2} \end{bmatrix} \right) E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} \quad (2.54)$$

which further reduces to

$$E\begin{bmatrix} w_{0k+1} \\ w_{1k+1} \end{bmatrix} = E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} + 2u\,E\begin{bmatrix} d_k x_k \\ d_k x_{k-1} \end{bmatrix} - 2u\,E\left(\begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} \begin{bmatrix} p_0 x_k + p_1 x_{k-1} \\ p_0 x_{k-1} + p_1 x_{k-2} \end{bmatrix}^T \right) E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} \quad (2.55)$$

Because X_k is a white zero mean sequence with unit variance, x_k is uncorrelated with x_{k-1} and Eq(2.55) becomes

$$E\begin{bmatrix} w_{0k+1} \\ w_{1k+1} \end{bmatrix} = E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} + 2u\,E\begin{bmatrix} d_k x_k \\ d_k x_{k-1} \end{bmatrix} - 2u\begin{bmatrix} p_0 & 0 \\ p_1 & p_0 \end{bmatrix} E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix}$$

$$= E\begin{bmatrix} w_{0k} \\ w_{1k} \end{bmatrix} + 2u\,E\begin{bmatrix} d_k x_k \\ d_k x_{k-1} \end{bmatrix} - 2u\,E\begin{bmatrix} p_0 w_{0k} \\ p_1 w_{0k} + p_0 w_{1k} \end{bmatrix} \quad (2.56)$$

In order to determine whether Eq(2.56) can converge to the optimum weight vector, the optimum weight vector is substituted into Eq(2.56). Using Eq(2.48) and Cramer's rule to calculate R^{-1} the optimum weight vector is given by

$$W^* = R^{-1}K^T = \begin{bmatrix} \dfrac{-(p_0^2+p_1^2)}{(p_0p_1)^2-(p_0^2+p_1^2)^2} & \dfrac{p_0p_1}{(p_0p_1)^2-(p_0^2+p_1^2)^2} \\ \dfrac{p_0p_1}{(p_0p_1)^2-(p_0^2+p_1^2)^2} & \dfrac{-(p_0^2+p_1^2)}{(p_0p_1)^2-(p_0^2+p_1^2)^2} \end{bmatrix} \times \begin{bmatrix} p_0 d_k x_k + p_1 d_k x_{k-1} \\ p_0 d_k x_{k-1} + p_1 d_k x_{k-2} \end{bmatrix} \quad (2.57)$$

which further reduces to

$$\mathbf{W}^* = \begin{bmatrix} \dfrac{-(p_0^2+p_1^2)(p_0 d_k x_k + p_1 d_k x_{k-1}) + p_0 p_1 (p_0 d_k x_{k-1} + p_1 d_k x_{k-2})}{(p_0 p_1)^2 - (p_0^2+p_1^2)^2} \\ \dfrac{p_0 p_1 (p_0 d_k x_k + p_1 d_k x_{k-1}) - (p_0^2+p_1^2)(p_0 d_k x_{k-1} + p_1 d_k x_{k-2})}{(p_0 p_1)^2 - (p_0^2+p_1^2)^2} \end{bmatrix} \quad (2.58)$$

For a minimum phase plant, the desired signal does not need to be delayed since the minimum phase plant's inverse is causal. Therefore, d_k is set equal to x_k and Eq(2.58) becomes

$$\mathbf{W}^*_{min} = \begin{bmatrix} \dfrac{-(p_0^2+p_1^2)p_0}{(p_0 p_1)^2 - (p_0^2+p_1^2)^2} \\ \dfrac{p_0^2 p_1}{(p_0 p_1)^2 - (p_0^2+p_1^2)^2} \end{bmatrix} \quad (2.59)$$

which is the optimum weight vector for minimum phase plants. Substitution of Eq(2.59) into Eq(2.56) yields

$$\lim_{k\to\infty} E[\mathbf{W}_{k+1}] = \mathbf{W}^* + 2u \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 2u \begin{bmatrix} \dfrac{-(p_0^4+p_1^2 p_0^2)}{(-p_0^4-p_0^2 p_1^2+p_1^4)} \\ \dfrac{-p_0 p_1^3}{(p_0 p_1)^2 - (p_0^2+p_1^2)^2} \end{bmatrix} \quad (2.60)$$

For a zero near the center of the z plane, $p_1/p_0 \ll 1.0$, the weight update equation Eq(2.60) can converge to the optimum weight vector

$$\lim_{k\to\infty} E[\mathbf{W}_{k+1}] = \mathbf{W}^* \quad (2.61)$$

since p_1^4 and p_1^3 are approximately zero. For a zero just inside the z plane unit circle, p1 /p0 < 1.0 , Eq(2.60) can not converge exactly to the optimum weight vector; so, the MSE error will exceed the minimum MSE given by Eq(2.49). Figure 7 shows the minimum MSE for the optimum weight vector $\mathbf{W}^*$ and the MSE performance for the weight vector $\mathbf{W}$ given by Eq(2.56) as function of p_1 with p_0 hardwired to 1. Since p_0 is set to 1, $-p_1$ is the zero of the plant. In Figure 7, the curve for $\mathbf{W}$ was generated by solving Eq(2.56) for $\mathbf{W}_k$ where $\mathbf{W}_k = \mathbf{W}_{k+1}$ and substituting $\mathbf{W}_k$ into performance surface function Eq(2.46). The $\mathbf{W}^*$ curve was generated by substituting $\mathbf{W}^*$ into Eq(2.49). The $\mathbf{W}$ curve shows that the performance degrades as the zero of the plant approaches the unit circle. The performance degradation is the result of the increasing phase response of the plant as the zero moves closer to the unit circle. The phase responses for plants with zeros at .1, .3, .5, and .9 are illustrated in Figure 8.

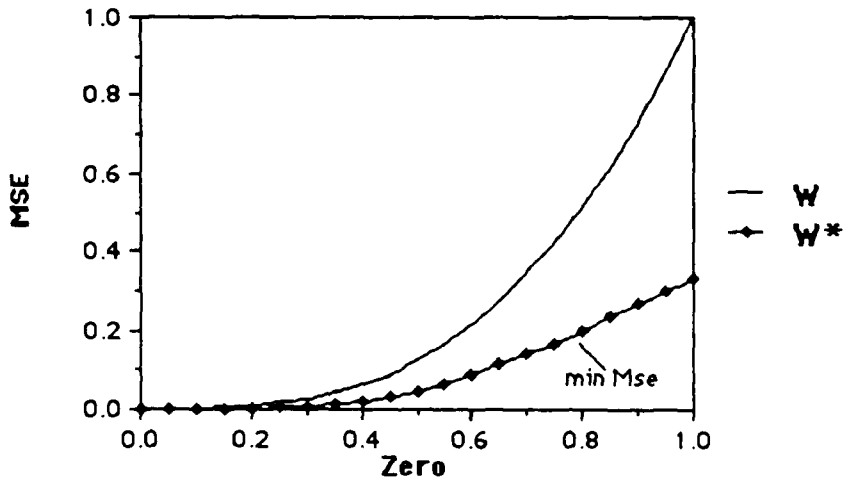


Figure 7. MSE versus the Zero Position for a Minimum Phase Plant

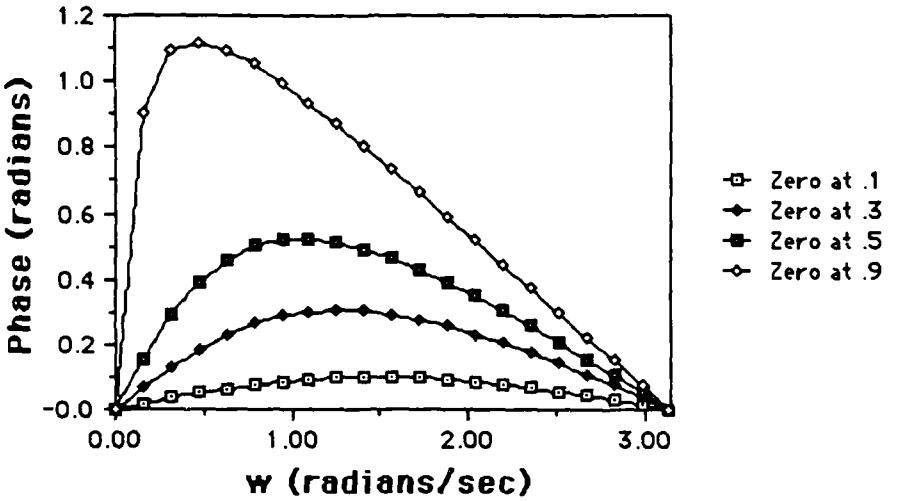


Figure 8. Plant Phase Responses

The inverse of non-minimum phase plant with all its zeros outside the unit circle has a left sided, non-causal, impulse response. To realize a causal inverse, the inverse is delayed to shift the left sided, non-causal impulse response to the right. The delay of the inverse is accomplished by delaying the desired signal. Thus, d_k is equal to x_{k-2} and Eq(2.58) becomes

$$W^*_{nonmin} = \begin{bmatrix} \frac{p_1^2 p_0}{(p_0 p_1)^2-(p_0^2+p_1^2)^2} \\ \frac{-p_1(p_0^2+p_1^2)}{(p_0 p_1)^2-(p_0^2+p_1^2)^2} \end{bmatrix} \qquad (2.62)$$

which is the optimum weight vector for a non-minimum phase plant.

Substitution of Eq(2.62) into Eq(2.56) gives

$$\lim_{k\to\infty} E[\mathbf{W}_{k+1}] = \mathbf{W}^* + 2u\begin{bmatrix}0\\0\end{bmatrix} - 2u\begin{bmatrix}\dfrac{p_0^2p_1^2}{(p_0p_1)^2-(p_0^2+p_1^2)^2}\\[2ex]\dfrac{-p_0^3p_1}{(p_0p_1)^2-(p_0^2+p_1^2)^2}\end{bmatrix} \qquad (2.63)$$

Equation (2.63) can not converge to the optimum weight vector when p_0 or p_1 are non-zero values. Figure 8 shows the MSE as given by Eq (2.46) for the optimum weight vector $\mathbf{W}^*$ and the weight vector $\mathbf{W}$ derived from Eq (2.56). The curve for $\mathbf{W}$ was generated by solving Eq(2.56) for $\mathbf{W}_k$ where $\mathbf{W}_k = \mathbf{W}_{k+1}$ and substituting $\mathbf{W}_k$ into performance surface function Eq(2.46). The $\mathbf{W}^*$ curve was generated by substituting $\mathbf{W}^*$ as given by Eq(2.62) into Eq(2.49). The MSE for $\mathbf{W}$ curve is unity for $\mathbf{W}$ given by Eq(2.56). Therefore, the AIP with the Eq(2.56) weight update equation will yield an irrelevant weight vector solution for a non-minimum phase plant with a single zero.

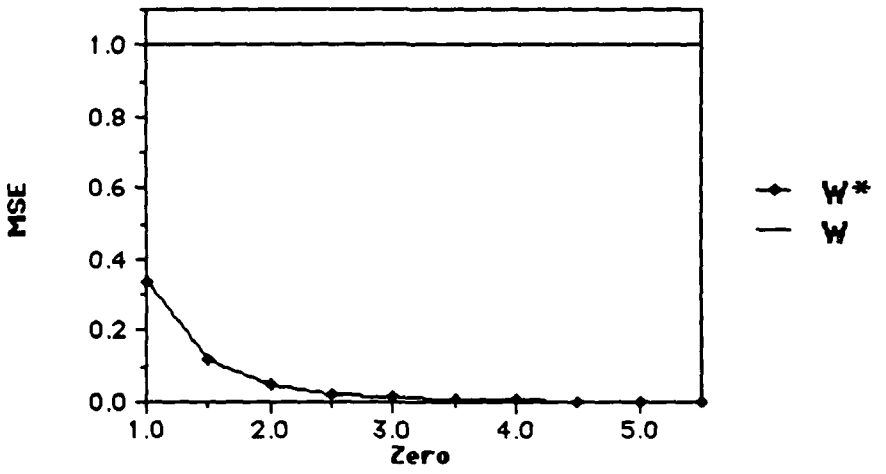


Figure 9. MSE versus the Zero Position for a Non-minimum Phase Plant

This example illustrates that a non-minimum phase plant decorrelates the LMS weight update inputs y'_k and X_k. Therefore, the AIP configuration will not find a relevant inverse weight vector when a non-minimum phase plant or a plant transport delay decorrelates the LMS weight update inputs. Since the audio plant is non-minimum phase and has a transport delay between the speakers and the microphone, the AIP is not a practical solution for this thesis. For a minimum phase plant, a relevant stable inverse can be obtained; however the performance is dependent on the phase response of the plant. For a minimum phase plant with a small phase response, y'_k and X_k are partially correlated and the weight update equation can converge to a relevant solution. AIP simulation results for minimum phase plants and non-minimum phase plants are provided in Chapter III which verify the theoretical results of this chapter.

By filtering the X_k in Eq(2.12) through a model of the plant to correlate the LMS inputs, y'_k and X_k, the AIP configuration can converge to a relevant weight vector solution. This filter configuration is called the filtered-x algorithm and, it is discussed in the next section.

Filtered-x Algorithm. The filtered-x algorithm compensates for the decorrelation effects of the plant as discussed in the AIP section by pre-filtering the LMS x_k signal through a model of the plant as shown in Figure 10. It will now be shown that the filtered-x weight update equation can converge to a relevant weight vector for both minimum and non-minimum phase plants.

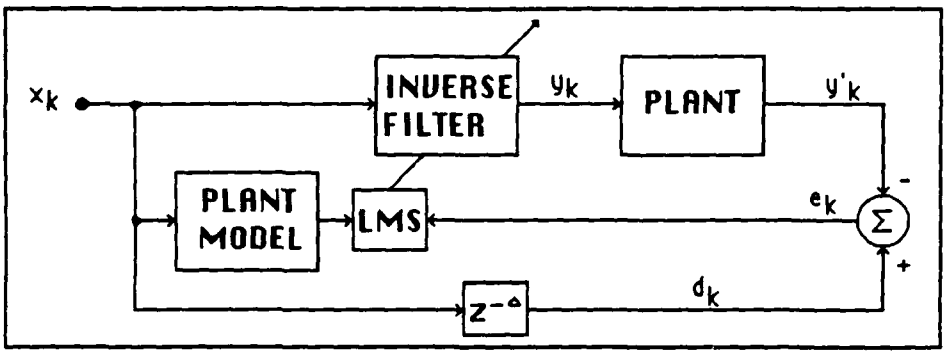


Figure 10. Filtered-x Algorithm (15:187)

The modified weight update equation which incorporates the filtering of X_k is given by (15:187)

$$W_{k+1} = W_k + 2ue_k\chi_k P' \tag{2.64}$$

where P' is the plant model vector. Substitution of Eq(2.41) into Eq(2.64) yields

$$W_{k+1} = W_k + 2u(d_k - P^T\chi_k W)\chi_k P' \tag{2.65}$$

The expected value of Eq(2.65) is

$$E[W_{k+1}] = E[W_k] + 2uE[(d_k - P^T\chi_k W)\chi_k P'] \tag{2.66}$$

Equation (2.64) can be rewritten as (15:187)

$$W_k = W_0 + 2u \sum_{j=0}^{k-1} e_j \chi_j P' \tag{2.67}$$

Since χ_j is filtered by the plant model P', the weight update input $\chi_j P'$ is no longer white and $\chi_k P'$ is correlated with $\chi_{k-1} P'$. Thus, W_k is not only correlated with $\chi_{k-1} P', \chi_{k-2} P' \ldots, \chi_0 P'$, but it is also correlated with $\chi_k P'$. The correlation between W_k and $\chi_k P'$ is proportional to the convergence constant u (15:187). For a very small u, W_k and $\chi_k P'$ are essentially uncorrelated. Assuming W_k and $\chi_k P'$ are uncorrelated by using a small u and the plant model is exact, P = P', Eq(2.66) can be expressed as

$$E[W_{k+1}] = E[W_k] + 2uE[d_k \chi_k P] - 2uE[\chi_k P P^T \chi_k^T] E[W_k] \quad (2.68)$$

Substituting Eqs(2.44) and (2.45) into Eq(2.68) yields

$$E[W_{k+1}] = E[W_k] + 2u(K^T - R\, E[W_k]) \quad (2.69)$$

When W_k is equal to W^*, Eq(2.69) becomes

$$E[W_{k+1}] = W^* + 2u(K^T - R\, W_k^*) \quad (2.70)$$

Substituting Eq(2.48) into Eq(2.70) gives

$$E[W_{k+1}] = W^* + 2u(K^T - R\, R^{-1} K^T)$$
$$E[W_{k+1}] = W^* \quad (2.71)$$

Thus, the filtered-x algorithm is capable of converging to the optimum weight vector for both minimum and non-minimum phase plants.

If the plant model is not known a priori, the plant can be directly modeled with an adaptive forward modeling filter as shown in Figure 11 (14:293). The adaptive forward modeling filter weights

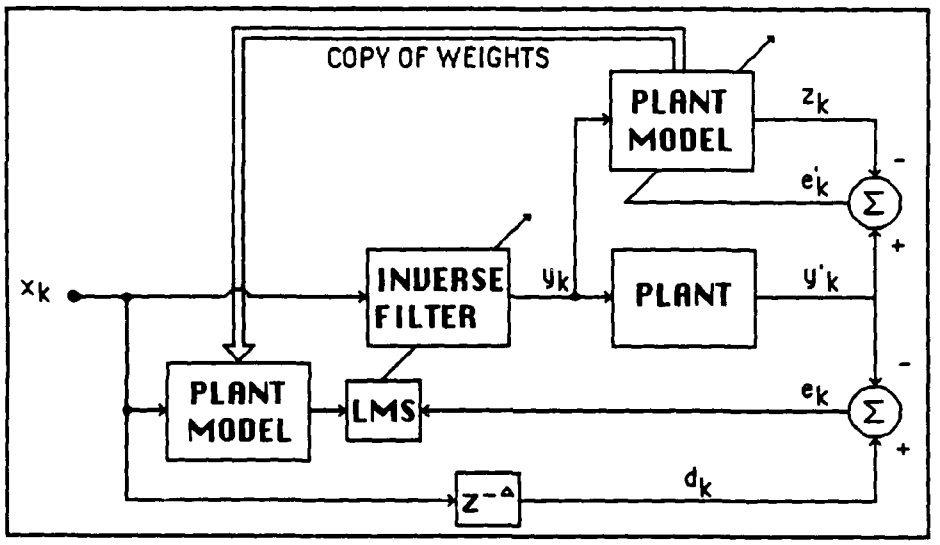


Figure 11. Filtered-x Algorithm with an Adaptive Plant Model (14:293)

are copied by the plant model filter to filter x_k for the LMS weight update. Experience has shown that the plant model does not have to be very precise (14:292). The plant model should have as least as great a transport delay as the the plant; so, the LMS inputs are correlated. The delays $z^{-\Delta}$ in Figures 10 and 11 are required for the non-minimum phase plant. The delays allow realizable causal approximation of a left sided or two sided non-causal impulse response. The last inverse filter discussed is the adaptive inverse modeling control system (AIMCS).

Adaptive Inverse Model Control System. The adaptive inverse model control system (AIMCS) is shown in Figure 12. The AIMCS was developed to inverse filter either non-minimum or a minimum

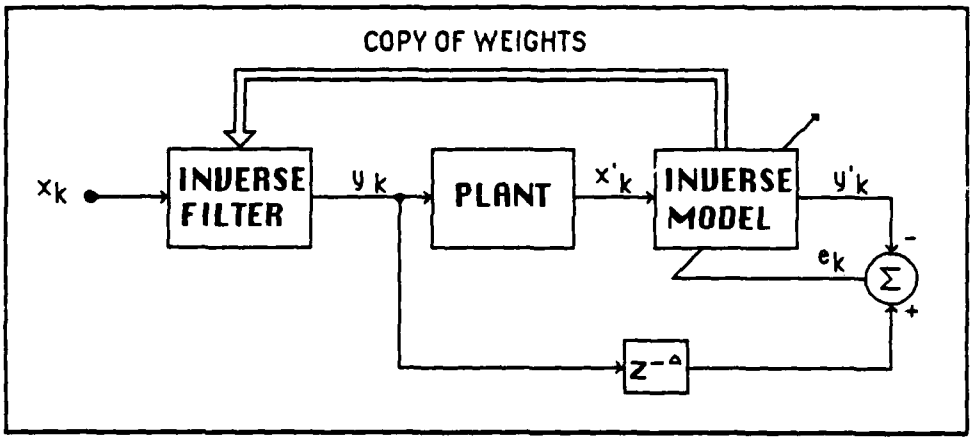


Figure 12. Adaptive Inverse Modeling Control System (14:281)

phase plants (14:280-285; 16:90-94). The AIMCS consist of an adaptive inverse modeling filter and the adaptive inverse filter which are shown in Figure 12. The adaptive inverse modeling filter in Figure 12 adapts its weights to cause its output to be a best least squares fit to the plant input. The weights of the adaptive inverse modeling filter are copied by the adaptive inverse filter which pre-filters the input x_k, so that the plant output x'_k is equal to x_k.

For a non-minimum phase, FIR plant, the optimal transfer function $W^*(z)$ of the adaptive modeling filter is infinite, two sided or left sided, and is given by (14:235)

$$W^*(z) = \frac{z^{-\Delta}}{P(z)} \qquad (2.72)$$

The weights from the adaptive modeling filter are transferred to the adaptive inverse filter W'(z) ; so, that

$$W'(z) = W^*(z) = \frac{z^{-\Delta}}{P(z)} \tag{2.73}$$

The adaptive inverse filter in cascade with the plant gives

$$W'(z)\,P(z) = \frac{z^{-\Delta}}{P(z)}P(z) = z^{-\Delta} \tag{2.74}$$

Thus, the output x_k' in Figure 1 1 is equal to $x_{k-\Delta}$, a delayed x_k.

For practical implementations where the inverse filter and the inverse modeling filter lengths are finite, the adaptive inverse filter approximates the perfect inverse. The Wiener weight vector for the Figure 1 2 AIMCS configuration is

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P} \tag{2.75}$$

where the input autocorrelation matrix $\mathbf{R}$ is given by

$$E[X_k X_k^T] = \mathbf{R} = E\begin{bmatrix} x'^2_k & x'_k x'_{k-1} & \cdots & x'_k x'_{k-L} \\ x'_{k-1}x'_k & x'^2_{k-1} & \cdots & x'_{k-1}x'_{k-L} \\ \vdots & \vdots & \ddots & \vdots \\ x'_{k-L}x'_k & x'_{k-L}x'_{k-1} & \cdots & x'^2_{k-L} \end{bmatrix} \tag{2.76}$$

and the cross correlation vector P between the input and the desired signal is given by

$$E[d_k \mathbf{X}_k^T] = P = E\begin{bmatrix} d_k x'_k \\ d_k x'_{k-1} \\ \cdot \\ \cdot \\ d_k x'_{k-L} \end{bmatrix} \qquad (2.77)$$

, and d_k is $y_{k-\Delta}$ (14:22). Unlike the AIP, the AIMCS can converge to a relevant inverse solution for both minimum and non-minimum phase plants because the plant does not decorrelate the LMS inputs. In addition, the AIMCS requires only two major components: an adaptive inverse modeling filter and an FIR filter with adjustable weights while the filtered-x requires three major components: an adaptive plant modeling filter, an adaptive inverse filter, and FIR filter with adjustable weights. The AIMCS is the primary time domain filter candidate for this thesis.

In summary, this chapter introduced the adaptive linear combiner and the LMS algorithm and developed inverse filter theory applicable to this thesis. The AIP, filtered-x algorithm, and the AIMCS inverse filter candidates were discussed. The next chapter presents the simulation results with the AIP, filtered-x algorithm, and the AIMCS inverse filter configurations.

III. Results and Discussion

This chapter presents and analyzes the results of experiments and computer simulations to verify the theory developed in Chapter 2 and to determine the feasibility of applying an inverse control system to remove the unwanted distortion effects of an audio system. The chapter begins with the results of computer simulations that show the AIP can inverse filter minimum phase plants while the filtered-x algorithm and the AIMCS can inverse filter both minimum and non-minimum phase plants. These initial simulations are followed by an analysis of the digitized data recorded at the AMRL/BBA reverberation chamber to demonstrate that the combination of the audio equipment and the reverberation chamber, which is designated the audio plant, distorts the input signal and that the power spectral response of the audio equipment and the reverberation chamber changes with time. The digitized data is then utilized to test the performance of simulated inverse models and inverse filters.

All computer program source code listings, which are referenced throughout this chapter by number, (i.e. (Prog 1)) are included in Appendix A.

Theory Verification Simulations

These simulations verify that the AIP can only inverse filter minimum phase plants while the filtered-x algorithm and the AIMCS

filter can inverse filter minimum and non-minimum phase plants. A three tap FIR filter with the difference equation

$$y_k = a_0x_k + a_1x_{k-1} + a_2x_{k-2} \tag{3.1}$$

was utilized as the plant where a_0, a_1, and a_2 are the filter coefficients. The z transform of Eq(3.1) is given by

$$Y(z) = a_0X(z) + a_1z^{-1}X(z) + a_2z^{-2}X(z) \tag{3.2}$$

The zeros of Eq(3.2) are selected to model either a minimum phase plant or a non-minimum phase plant.

For the first simulations, a minimum phase plant with a pole and zeros as shown in Figure 13 was inversed filter by the AIP, filtered-x algorithm and the AIMCS. In Figure 13, the pole is designated with an "X", and the zeros are designated with an "O". The corresponding phase response for

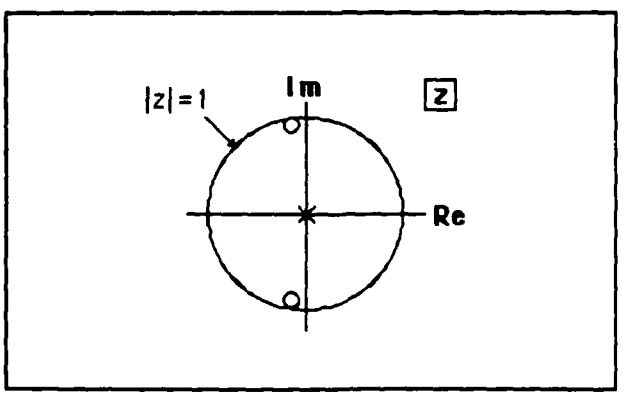


Figure 13. Plant Pole, Zero Plot

the Figure 13 pole zero combination is illustrated in Figure 14. The plant phase response never exceeds 1.2 radians. The learning curves for the AIP, filtered-x algorithm, and the AIMCS are shown in Figures 15, 16, and 17, respectively (Progs 1,2,3). The adaptive filter length for all three cases is 21 taps. The large number of taps, in comparison to the 3 plant taps, is required since the inverse of the FIR plant model has an infinite impulse response. The 21 tap FIR inverse approximates the infinite impulse response. Each learning curve is the average of 100 runs with a different random number generator seed. The learning curves' ordinate is the Mean Square Error, $E(e^2)$, and the abscissa is the iteration number. All three inverse filter configurations converged to a relevant stable inverse filter solution.

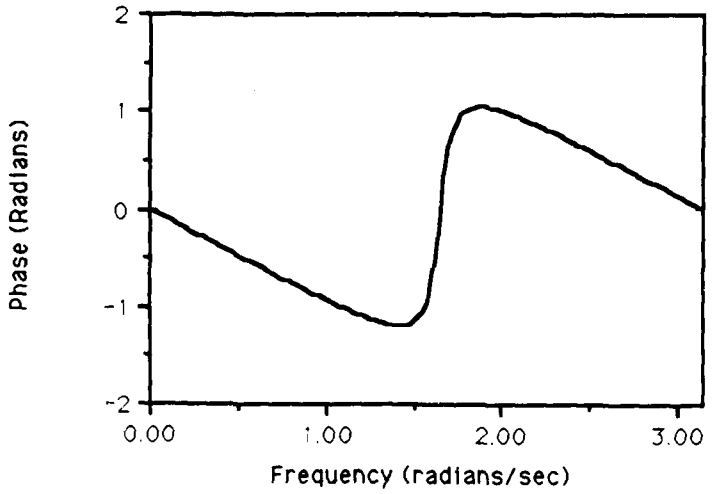


Figure 14. Plant Minimum Phase Phase Response

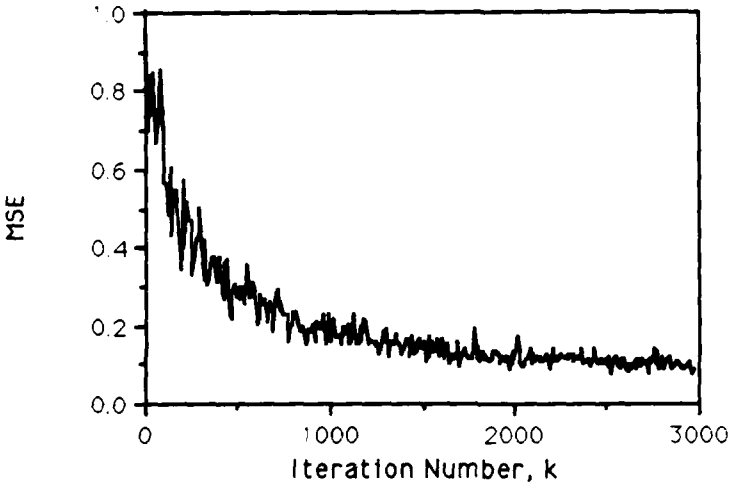


Figure 15. AIP Learning Curve for a Minimum Phase Plant

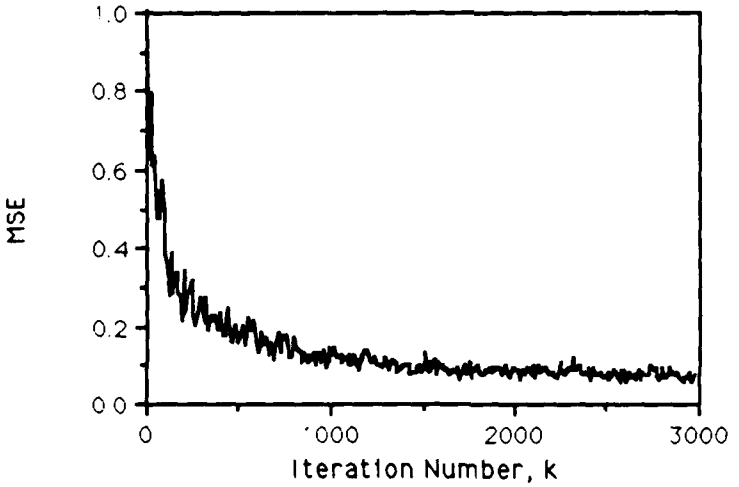


Figure 16. Filtered-x Learning Curve for a Minimum Phase Plant

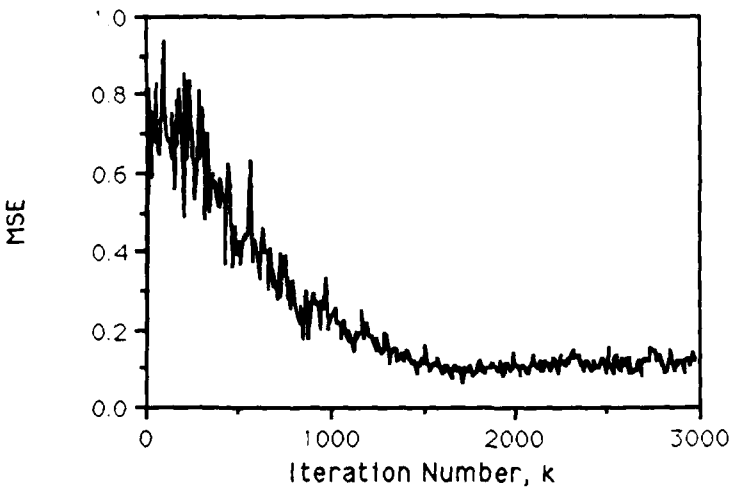


Figure 17. AIMCS Learning Curve for a Minimum Phase Plant

A non-minimum phase plant was also modeled with a three tap FIR filter. The pole zero plot and the corresponding phase response for the non-minimum phase plant are illustrated in Figures 18 and 19, respectively. Since there are two zeros outside of the unit circle, the phase is 2π radians at π radians per second which agrees with the non-minimum phase theory discussed in Chapter 2.

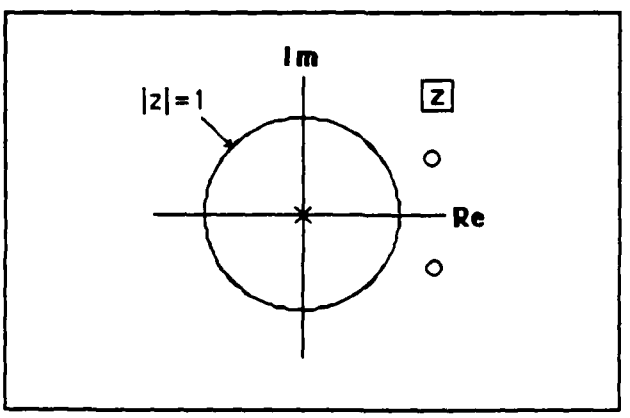


Figure 18. Plant Pole, Zero Plot

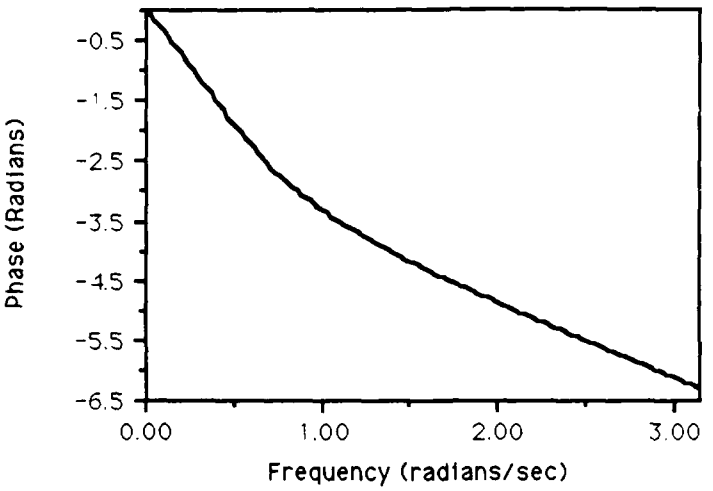


Figure 19. Non-minimum Phase Plant Phase Response

The learning curves for AIP, filtered-x algorithm, and the AIMCS are shown in Figures 20, 21, and 22, respectively. Unlike the filtered-x and the AIMCS, the AIP does not reach a relevant inverse filter solution in 3000 iterations, which is evident in Figure 20. In fact as the number of iterations increases, the AIP MSE exceeds $E[d_k^2]$ since the AIP weight update equation for this example does not converge to an all zero weight vector solution. A 30000 iteration learning curve for the AIP is shown in Figure 23. The excessive MSE was still exhibited when the AIP simulation was repeated with smaller convergence constants. As discussed in Chapter 2, the inability of the AIP to find a relevant inverse weight vector is the consequence of the plant's large phase response which decorrelates the x_k and the y'_k LMS weight update inputs.

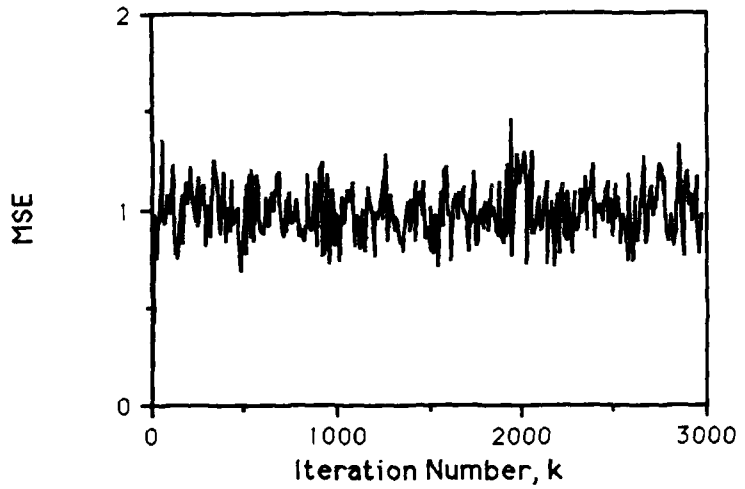


Figure 20. AIP Learning Curve for a Non-minimum Phase Plant

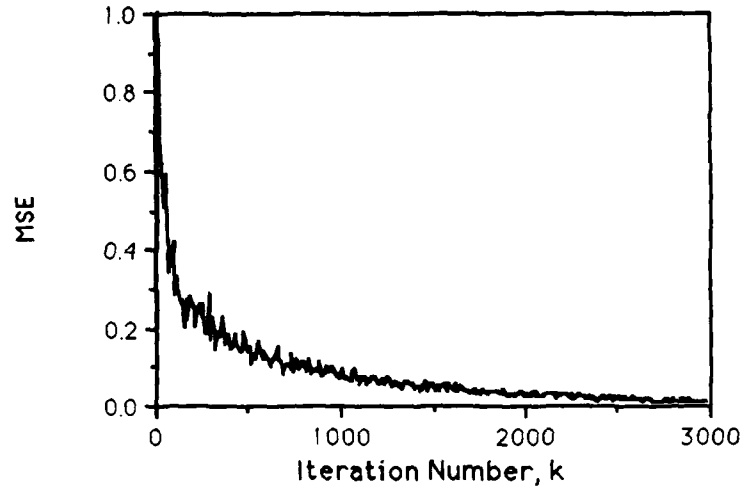


Figure 21. Filtered-x Learning Curve for a Non-minimum Phase Plant

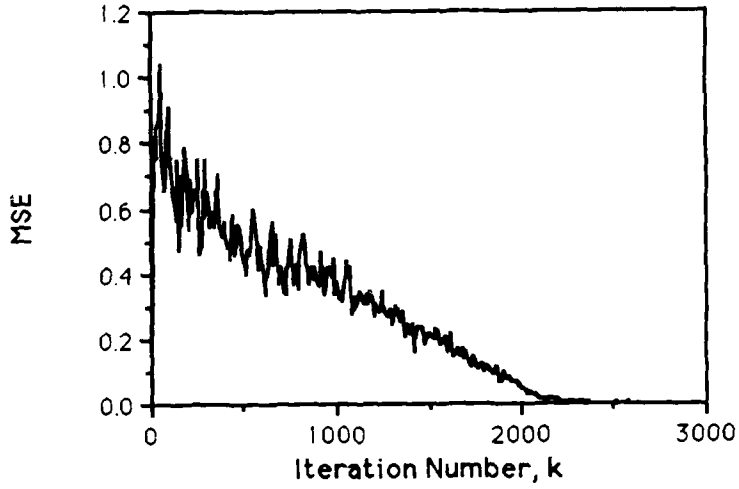


Figure 22. AIMCS Learning Curve for a Non-minimum Phase Plant

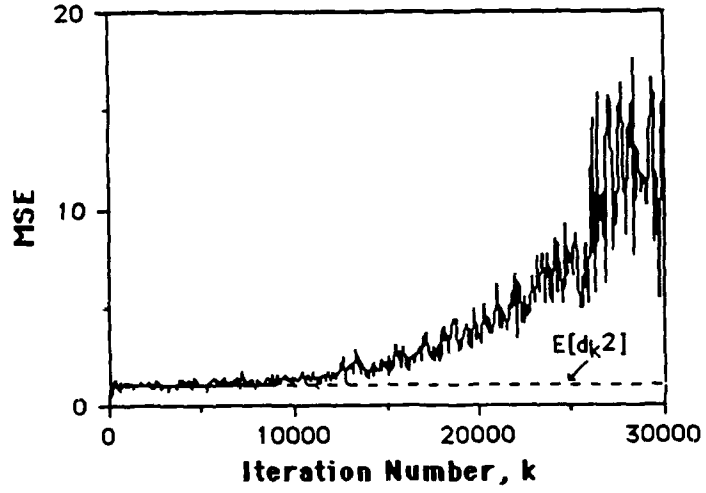


Figure 23. AIP Learning Curve Showing Excessive MSE for a Non-minimum Phase Plant

The predicted MSE performance versus the location of a zero for the two tap minimum phase plant and the two tap AIP as illustrated by the **W** curve in Figure 7 was verified by simulating the Figure 6 configuration (Prog 4). The simulation MSE along with the predicted and the minimum MSE curves are shown in Figure 24. The simulation MSE curve is in good agreement with the predicted curve except for a slightly lower overall MSE. The overall lower MSE for the simulation could be the result of some correlation between consecutive samples of the input pseudo white noise sequence. This lower overall MSE for the simulation will not be investigated further.

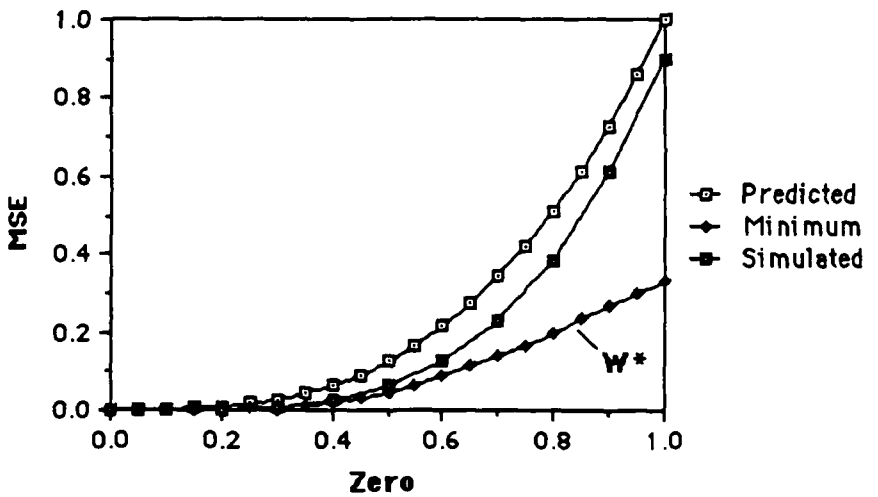


Figure 24. Simulated MSE versus the Zero Position for a Minimum Phase Plant

Audio Plant Analysis

This section begins with a brief description of the AMRL/BBA reverberation chamber's audio equipment. The digitized data from the reverberation chamber is then analyzed to determine if the response of the chamber varies with time, to determine if the input

audio signal is distorted by the plant, and to measure the impulse response of the chamber.

AMRL/BBA Reverberation Chamber Description. A block diagram and an approximate overhead view of the reverberation chamber is shown in Figure 25. The white noise generator generates zero mean

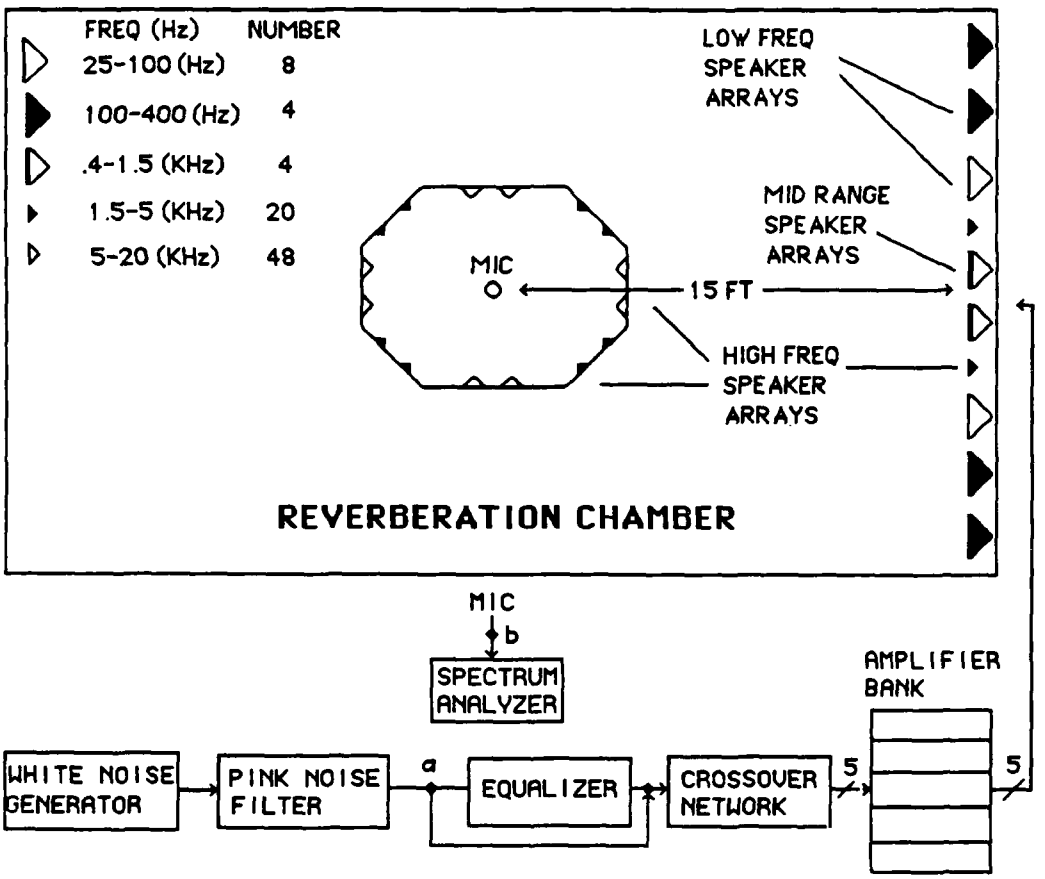


Figure 25. Audio System and Reverberation Chamber

noise with a selectable standard deviation. For the data collection, the standard deviation was set to 2 which is the normal setting for AMRL testing. The white noise is passed through a pink noise filter. The pink noise filter has a rolloff of 3 dB per octave from to 20 to

20000 Hz. The 32 band graphic equalizer in Figure 25 was not available and was bypassed during digitized data collection. The pink noise is filtered through a five way crossover network with passbands at 25-100 Hz, 100-400 Hz, 400-1500 Hz, 1500-5000 Hz, and 5000-20000 Hz. The crossover bands are amplified and distributed to the low frequency, wall mounted speaker arrays or the ceiling suspended high frequency array as shown in Figure 25. Located at the center of the reverberation chamber and suspended approximately 6 ft from the floor is a reference microphone (MIC). The reference microphone signal's spectrum is displayed on a spectrum analyzer which has a selectable averaging period. The graphic equalizer is adjusted to maintain a desired spectrum at the reference microphone.

The transfer function of the reverberation chamber is given by

$$M(e^{jw}) = N(e^{jw})\, P(e^{jw}) \qquad (3.3)$$

where $M(e^{jw})$ is the frequency response of the signal at the microphone, $N(e^{jw})$ is the frequency response of the pink noise, $P(e^{jw})$ is the frequency response of the audio plant which includes the frequency response of the crossover network, the power amplifiers, the speaker arrays, and the acoustic transmission paths between the speakers and the microphone.

Digitized Data Analysis. Digitized data from points a and b in Figure 25 was simultaneously recorded using a Hewlett Packard dual channel, 12 bit digitizing oscilloscope. To prevent aliasing, the input

to each oscilloscope was low pass filtered with a 5th order butterworth filter and sampled at approximately 3.5 times the audio bandwidth of 12 KHz. The actual A/D sampling rate of the oscilloscope was 43956 Hz which equates to an 22.75 microsecond sampling period supported by the digital oscilliscope. The digital oscilloscope does not have unlimited sampling rates since the sampling rates are derived from the oscilloscope's internal clock. The audio system's gain was adjusted so that the overall sound pressure level of the pink noise at the microphone pickup was 115 dB SPL. Each 12 bit A/D data sample was saved to the oscilloscope's 3.5 inch disk drive as two eight bit words. Data files were saved as a block of 16384 consecutive samples. Twenty-five data files were recorded: 12 pink noise files from point a, 12 pink noise files from point b, and 1 impulse response file from point b. The data files were translated by the oscilloscope to a four digit integer representation and transferred to a Zenith 248 through a IEEE 488 interface for storage on a transportable 5.25 inch MSDOS floppy disk. The MSDOS files were down-loaded into ASCII files on the AFIT's Elxsi supercomputer and the VAX 785 for subsequent processing.

The Power Spectral Densities (PSD) from points a and b in Figure 25 were first analyzed to determine if and how the audio plant distorts the input pink noise spectrum. An averaged periodogram estimator was utilized to calculate the PSDs. The mathematical expression for the averaged periodogram estimator is given by

$$H(\omega_k) = \frac{1}{K} \sum_{m=0}^{K-1} \frac{1}{L} \left| \sum_{n=0}^{L-1} x_m(n) \exp(-j\omega_k n) \right|^2 \qquad (3.4)$$

where $x_m(n)$ is the nth sample, ω_k is the kth radian frequency, L is the number of DFT points, and K is the number of non-overlapping blocks of length L (6:68). The data record of length N is segmented into K non-overlapping blocks of length L where N = K L. To reduce the PSD computation time, the DFT in Eq(3.4) was replaced with a radix 2 FFT. Figure 26 shows the PSDs at the output of the pink noise filter, point a, and at the microphone output, point b, where N = 32768, L = 256, and K =128. Since the frequencies of interest cover the range from 25 to 12000 Hz, only 70 of the 256 PSD harmonics were plotted. Figure 26 shows that the audio plant attenuates the overall desired pink noise spectrum especially at the higher frequencies. So, a manual or adaptive spectrum shaper is required to boost the attenuated frequencies.

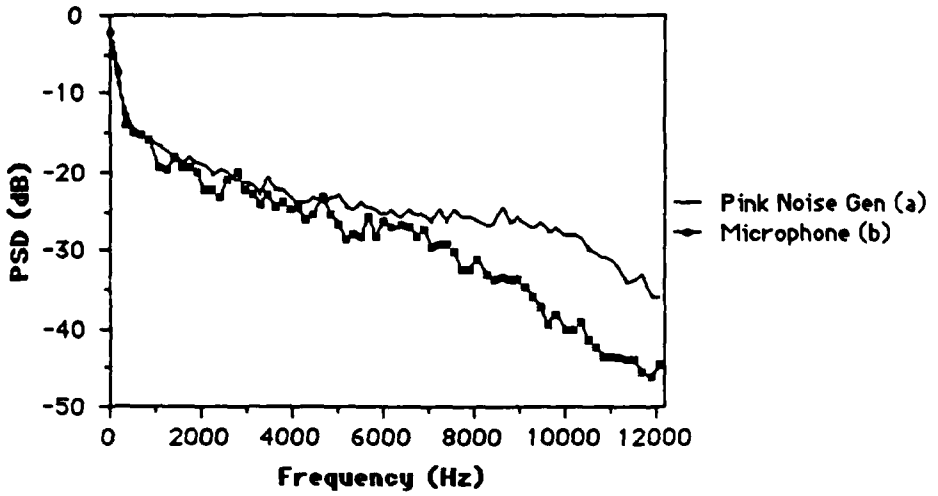


Figure 26. PSDs at the Pink Noise Input and Microphone Output;

To determine whether the frequency response of the audio plant changes with time, data were collected at 5 minute intervals, the PSDs were calculated, and the PSDs were compared. To compare the PSDs, the average PSD magnitude difference was calculated between the PSD at time t = 0 minutes and at time t = 5 n minutes where n is integer value. The average PSD magnitude difference is given by

$$\text{AvgDiff} = \frac{1}{P} \sum_{k=0}^{P-1} \left| 10\log(H_0(\omega_k)) - 10\log(H_n(\omega_k)) \right| \qquad (3.5)$$

where P is the number of harmonics averaged, $H_0(\omega_k)$ is the PSD at 0 minutes and $H_n(\omega_k)$ is the PSD at 5 n minutes. Figure 27 shows a graph of the average magnitude difference versus time for the microphone output and the pink noise generator. The curve for the pink noise generator shows

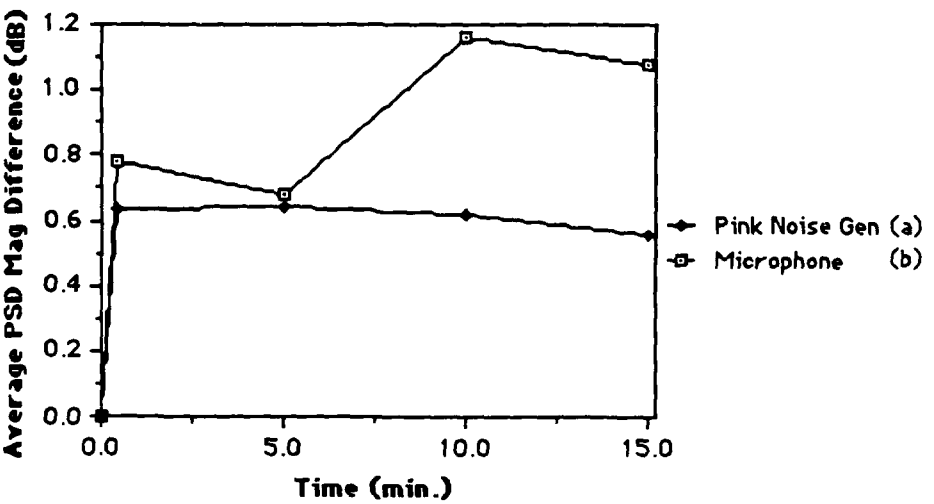


Figure 27. Average PSD Magnitude Difference versus Time;

that the pink noise input spectrum changed approximately .6 dB while the spectrum at the microphone output changed 1.2 dB after 15 minutes. This appears to verify that the frequency response of the audio plant varies slowly with time since the PSD at the microphone exhibited a larger change than the PSD at the pink noise generator. Because the audio plant's frequency response is non-stationary, an adaptive inverse system is highly desirable since the manual equalization should be repeated periodically.

Impulse Response Measurement. An acoustic impulse signal was generated at the room center approximately 3 ft from the floor and recorded at the microphone output. Figure 28 shows the impulse response of the reverberation chamber. The impulse response has a

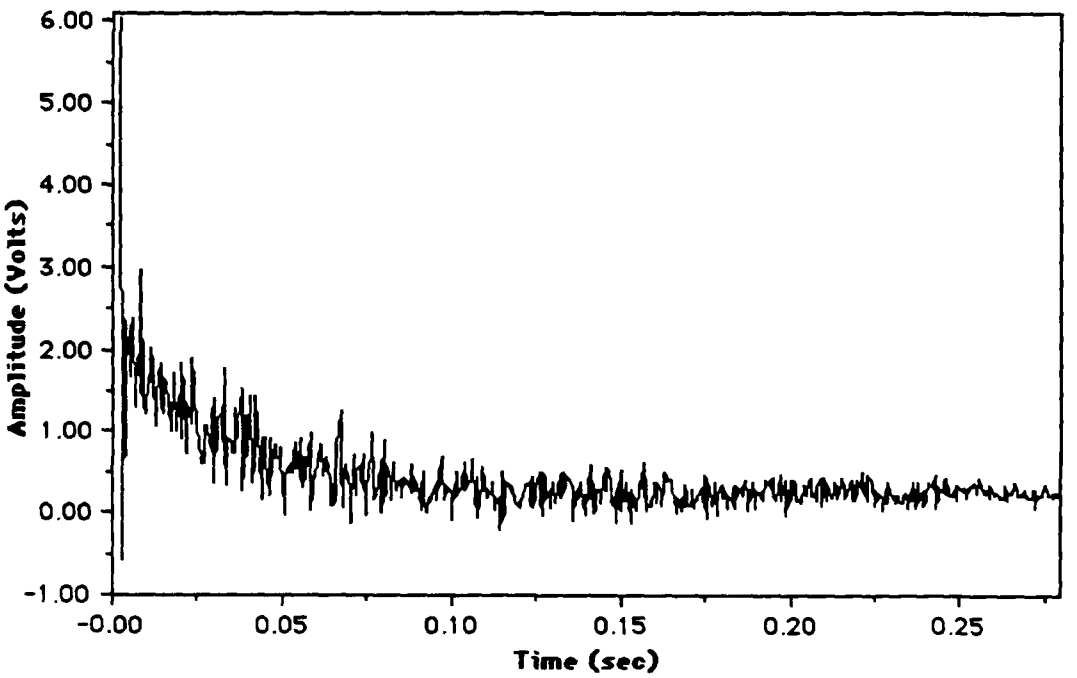


Figure 28. Reverberation Chamber Impulse Response

duration of nearly 300 milliseconds and does not include the impulse response of the crossover network, the audio amplifiers, or the speaker arrays. To accurately model the entire audio plant with an FIR model, the model's impulse response duration would have to exceed 300 milliseconds to account for both the reverberation chamber and the audio equipment.

Audio Plant Inverse Model and Filter Simulation Results

This section presents the inverse model and inverse filter simulations results for the audio plant. The overall objective of the simulations is to explore the feasibility of using an FIR adaptive filter for the automatic equalization of the audio plant which distorts the input pink noise spectrum. The primary performance criterion used to assess the merit of the simulated inverse model and inverse filter is a comparison of the desired pink noise spectrum with the spectrum at the audio plant model output for the inverse filter simulations and with the spectrum at the output of the adaptive inverse model for the inverse model simulations. All simulations are non-real time since dedicated digital signal processing hardware was not available.

The number of filter taps for the simulation inverse models and the inverse filters was limited to the number , which could be implemented, in real time with a high speed digital signal processor (DSP). AFAMRL has targeted the Texas Instruments TMS320C30 as the DSP for future hardware development. A 150 nanosecond per tap LMS adaptive filter weight update along with the FIR filter and data shift is a specified benchmark for the TMS320C30 DSP (10:537).

With the plant audio bandwidth (bw) of 25 Hz to 12 kHz and a sampling rate, fs, of 43956 Hz, which meets the Nyquist criteria of greater than 2bw, the maximum number of taps feasible with the TMS320C30 is 151 taps.

Adaptive Inverse Model. The first simulation results discussed is for the adaptive inverse model (AIM). The AIM is the adaptive inverse modeling component of the AIMCS and is shown as a subcomponent of the AIMCS in Figure 29. Figure 30 shows the AIM simulation block diagram. The digitized data from the pink noise

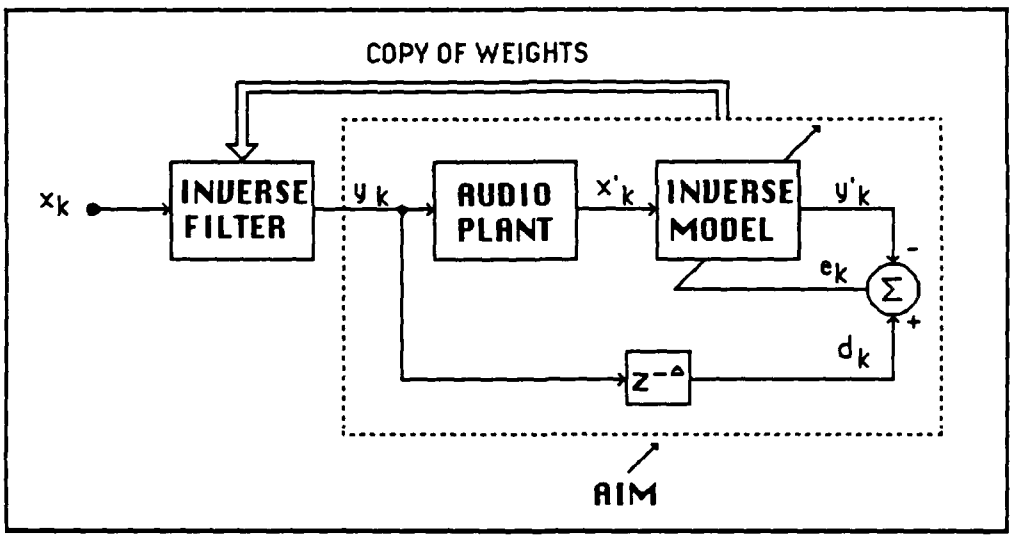


Figure 29. AIMCS with the AIM Component Identified

filter was the AIM desired sequence, and the digitized data from the microphone was the adaptive filter input signal. The PSDs at the points a' , b, and c are shown in Figure 31 for a 151 tap AIM (Prog 5). The average PSD magnitude difference was 4.8 dB for the a' and b curves and was 2.6 db for the a' and c curves. Ideally, the PSDs at

points a' and c would be equal and the average PSD magnitude difference would be 0 db if the AIM converges to the exact delayed inverse. As shown by Figure 31, the performance degrades at the higher frequencies.

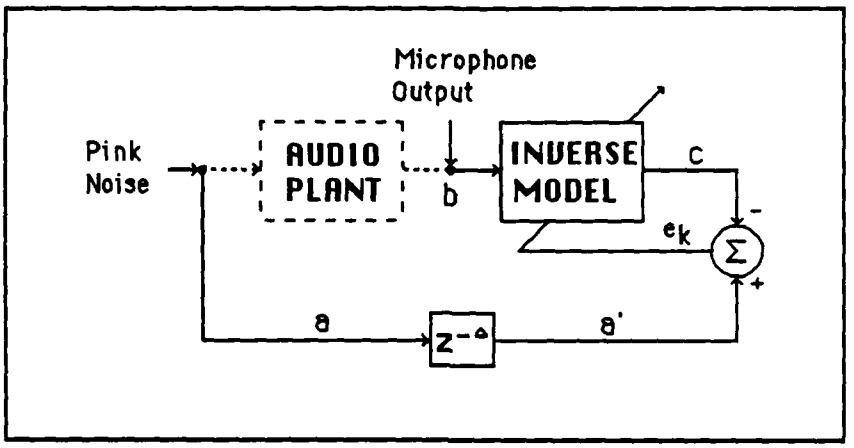


Figure 30. AIM Simulation Block Diagram

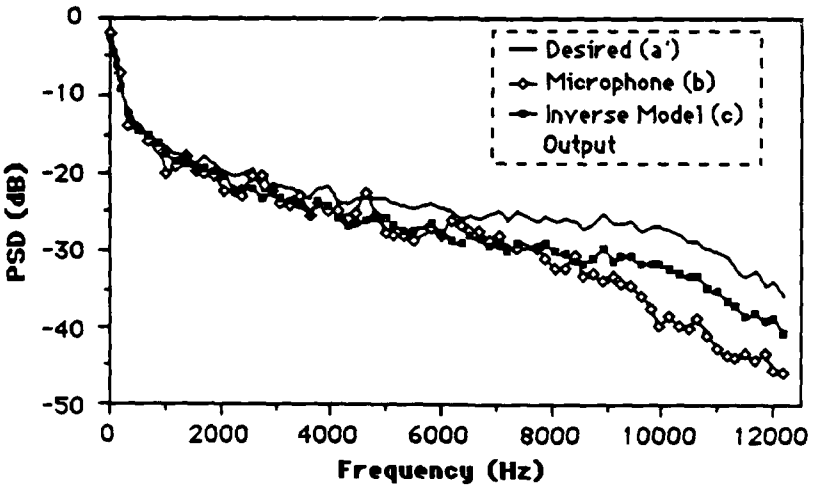


Figure 31. AIM Power Spectral Densities of the Desired Signal (a'), of the Audio Plant (b), and of the AIM output (c)

It was observed that the high frequency performance of the inverse model was a function of the convergence constant ,u . The maximum convergence constant possible, which still allowed stable operation, provided the best overall and high frequency inverse performance. The average PSD magnitude difference between a' and c as a function of convergence constant is shown in Figure 32. The number of iterations was increased for the smaller convergence constants to ensure the adaptive filter had converged before calculating the PSDs. The inferior performance at the higher frequencies is analyzed in the next paragraph.

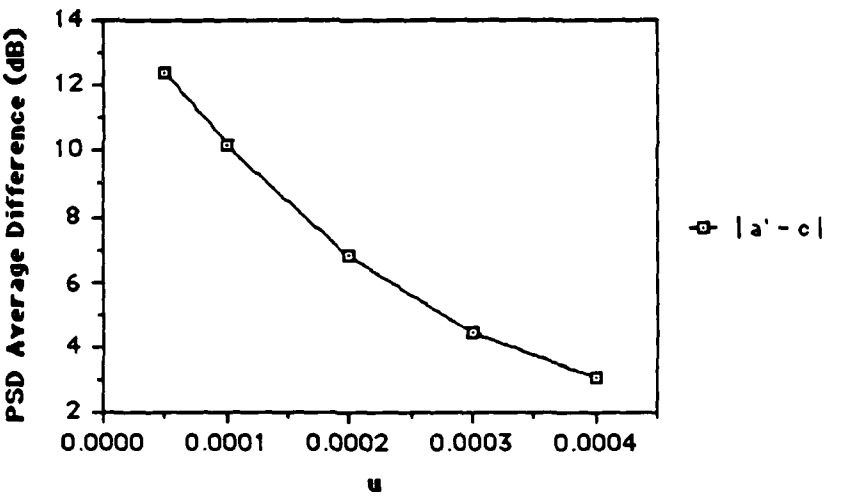


Figure 32. PSD Average Difference Versus u

The author's hypothesis is that the poor high frequency performance is the consequence of the fixed convergence constant, the fast sampling rate, the limited number of filter taps, and the pink noise spectrum. It was observed that the weights of the AIM never converged to a single optimum weight vector but were constantly

changing. The non-convergent weight behavior is attributed to the AIM input power variations. Recall that the optimum weight vector is given by $\mathbf{W}^* = \mathbf{R}^{-1}\,\mathbf{P}$ where $\mathbf{R}$ is the filter input correlation matrix. The sum of the diagonals elements of $\mathbf{R}$ is the AIM input power as seen by the adaptive inverse model at point b in Figure 30. The input power is constantly changing because of the small number of taps, the 43956 Hz sampling rate and the large low frequency components. A snapshot of the input trace at point b is shown in Figure 33. Two 151 sample windows are identified which illustrate the small size of the adaptive filter window in comparison to the large low frequency components.

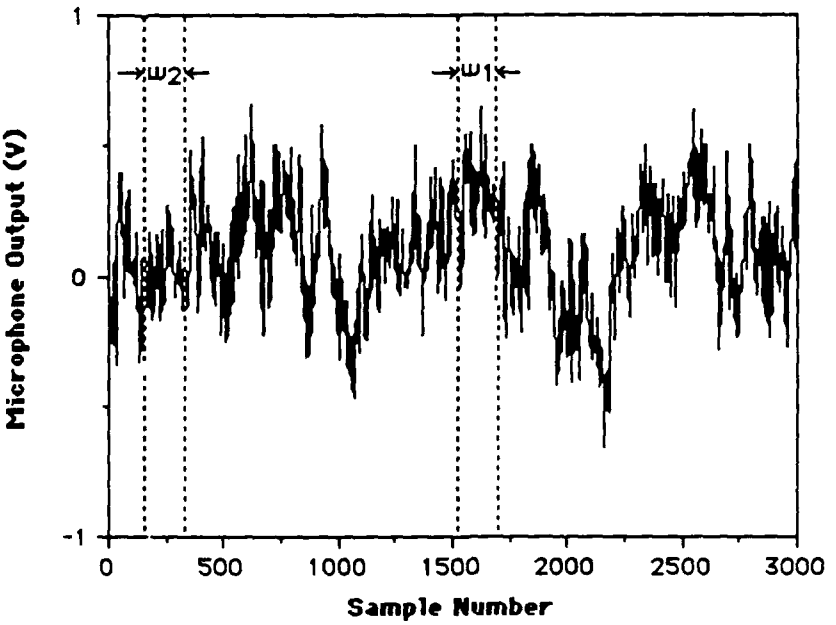


Figure 33. Microphone Output with Two151 Sample Windows

The input power of the kth iteration is given by

$$\text{Input Signal Power} = \sum_{n=0}^{N-1} x_{(k-n)}^2 \qquad (3.6)$$

where x_k is the kth input component and N is the number of taps. Figure 34 shows the input power at point b as a function of the sample number.

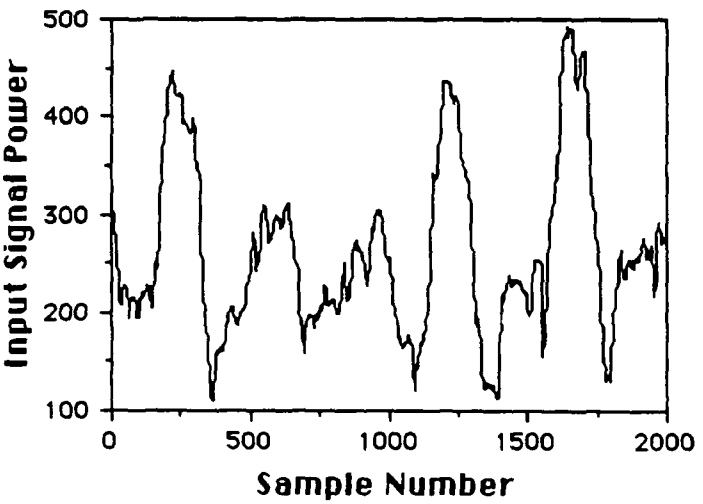


Figure 34. AIM Input Power as Seen by the 151 Tap Adaptive Inverse Model

Because of the large input power variations, the optimum weight vector $\mathbf{W}^*$ constantly changes. To minimize the weight misadjustment, $|\mathbf{W} - \mathbf{W}^*|$, the adaptive inverse model weight vector $\mathbf{W}$ must rapidly converge to and track the nonstationary optimum

weight vector $\mathbf{W}^*$. However, the speed of the LMS adaptation is limited by the size of the convergence constant u. The upper bound of the the convergence constant , which is inversely proportional to the input power, is given by (14:103)

$$u < \frac{1}{(L+1)(\text{Signal Input Power})} \tag{3.7}$$

where L is the number of taps. Therefore, the larger power low frequency components place a limit on u_{max}. Any attempt to make u larger to track the lower power high frequency components would cause instability. Thus, it could be concluded that the frequency response of an adaptive filter evolves fastest in bands of highest energy.

To improve inverse model performance at the higher frequencies and to support the above hypothesis, the normalized LMS (NLMS) was incorporated into the AIM simulation. The NLMS varies the convergence constant based on the input signal power to achieve continuous rapid adaptation. The expression for the NLMS convergence constant u(k) is given by (13:82-83)

$$u(k) = \frac{\alpha}{\gamma + \text{Input Signal Power}} \tag{3.8}$$

where α is a constant which is selected to achieve rapid convergence and γ is a small constant which prevents excessively large u(k) values when the input power is negligible. The PSDs at points a', b,

and c are shown in Figure 35 for the NLMS AIM simulation. The AIM high frequency inverse performance has been enhanced with the NLMS, and the a' and c PSD curves are now almost identical. The NLMS compensates for the input power variations by calculating the optimum convergence constant for the input power. The improved high frequency performance with the NLMS supports the above hypothesis.

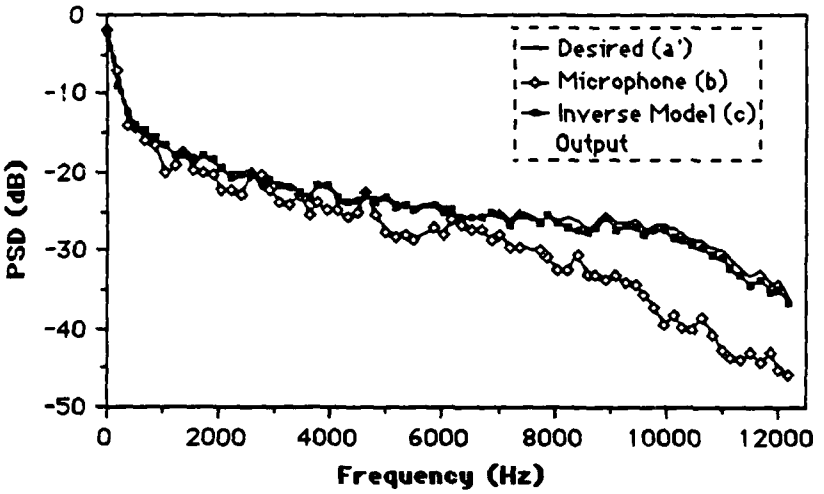


Figure 35. AIM/NLMS Power Spectral Densities of the Desired Signal (a'), of the Audio Plant (b), and of the AIM output (c)

The optimum value of the inverse delay, $z^{-\Delta}$, in Figure 30 was empirically determined to be 100 samples for a 151 tap filter. A plot of the PSD average magnitude difference between curves a' and c as a function of the delay is shown Figure 36.

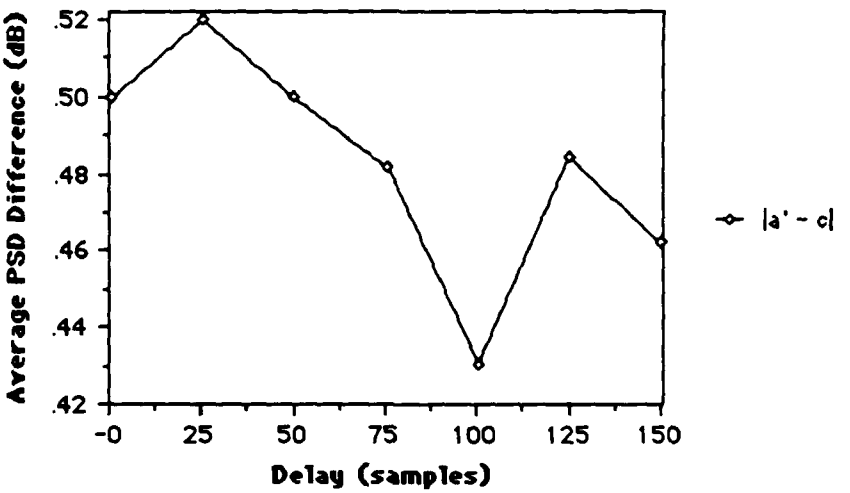


Figure 36. AIM/NLMS Performance versus Inverse Delay

Inverse Filter Simulations. The inverse filter simulations were conducted to determine if the AIMCS or the filtered-x algorithm could inverse filter the audio plant. The results of these simulations provided valuable insight for reaching the recommendation to implement the AIMCS or filtered-x in hardware or pursue an alternative approach.

Plant Model. In order to accomplish the AIMCS simulations, an audio plant model was required since the inverse filter stage of the AIMCS occurs forward and in series with the audio plant. The audio plant model weight vector was generated with an LMS adaptive forward model (AFM) (Prog 6). The AFM, which is shown in Figure 37, adapts its weights so that the AFM output is a least squares fit to the audio plant output (14:195-196).

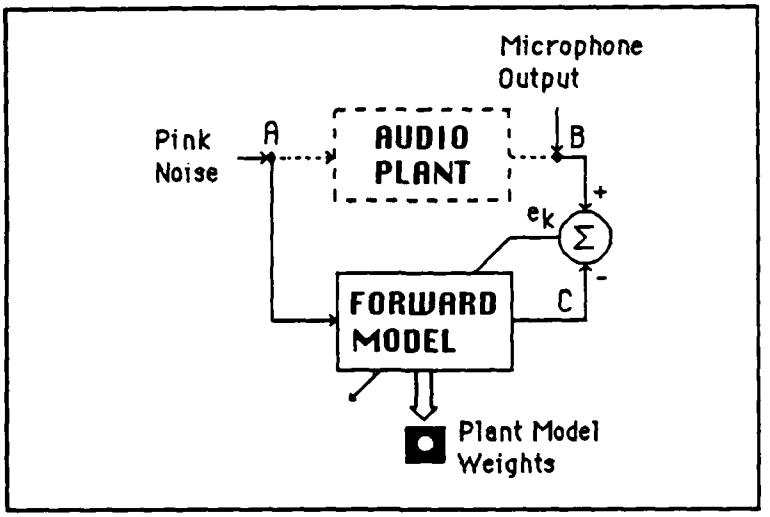


Figure 37. AFM Block Diagram

Since all the following simulations process blocks of the digitized data less than 1.5 seconds in duration, it has been assumed that the audio plant impulse response remains stationary. Figure 27 illustrated that the spectral response of the audio plant slowly varied and that the first apparent change occurred between 5 and 10 minutes. Because of the assumed stationary spectral behavior of the audio plant for the short duration of the simulations, only one weight vector was saved at the last iteration of the adaptive forward modeling process as the audio plant model.

Since the audio plant forward model will not be part of the AIMCS if the AIMCS is implemented in hardware, the number of taps is not limited to 151. More than 13000 FIR taps at the 43956 Hz sampling rate would be required for the audio plant model to match the duration of the reverberation chamber's impulse response shown in Figure 28. Since a 13000 tap forward model simulation is a

computational burden, the initial simulations used a 1000 tap FIR filter.

Figure 38 shows the PSDs at points A, B, and C for the 1000 tap, LMS, AFM simulation. The average magnitude difference between the PSDs of B and C was 1.6 dB and between A and B was 4.6 dB. Ideally the curves for B and C would match if the AFM and the audio plant impulse responses were identical.

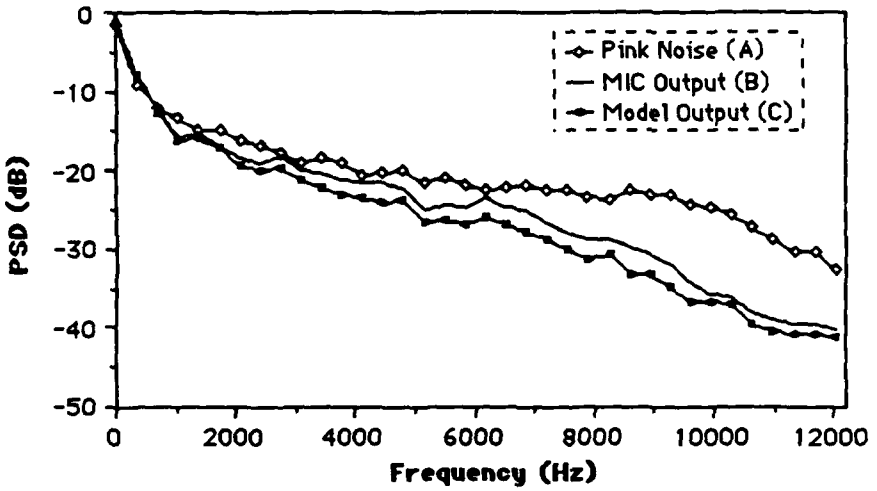


Figure 38. AFM Power Spectral Densities of the Pink Noise Input (A), Microphone Output (B), and the AFM Output (C)

Before proceeding with the AIMCS simulations, a NLMS AIM simulation was conducted with the audio plant model weight vector to verify that an inverse model could be generated for the audio plant model weight vector. The block diagram for the AIM simulation test is shown in Figure 39, and the corresponding PSDs are shown in Figure 40. The average magnitude difference between the

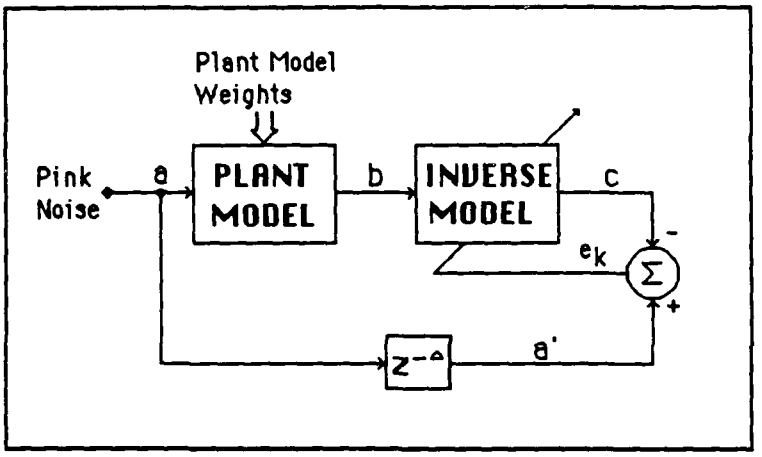


Figure 39. AIM with Plant Model

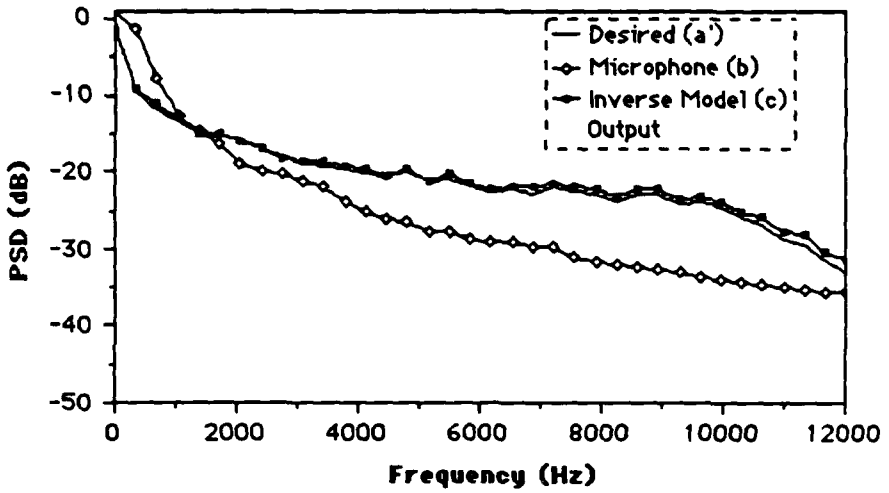


Figure 40. AIM and Plant Model Test's Power Spectral Densities of the Desired Signal (a'), of the Audio Plant (b), and of the AIM output (c)

PSDs of a' and c was .6 dB and between a and b was 5.2 dB. The AIM test with the plant model demonstrated that the AIM could still generate the inverse model for the plant model.

AIMCS Simulations Results. The AIMCS NLMS simulation block diagram is shown in Figure 41 (Prog 7). For this simulation, the objective of the AIMCS is to remove the distortion effect of the plant; so, the PSD at the output of the plant model, point C, matches the PSD of the desired pink noise spectrum, point A. The simulation PSDs curves for I/O points A and C are shown in Figure 42. The PSD curve at the output of the plant with the inverse filter bypassed, "Without Inverse", was also included in Figure 42 to allow a comparison of the plant model output PSDs with and without the inverse filter. Figure 42 shows that the AIMCS only improved the match in the frequency range of 0 to 350 Hz. In fact, the AIMCS had a deleterious effect at frequencies above 350 Hz. From 350 to 12000 Hz, the PSD at the plant output with the inverse filter bypassed matched the desired pink noise spectrum better than when the inverse filter was enabled.

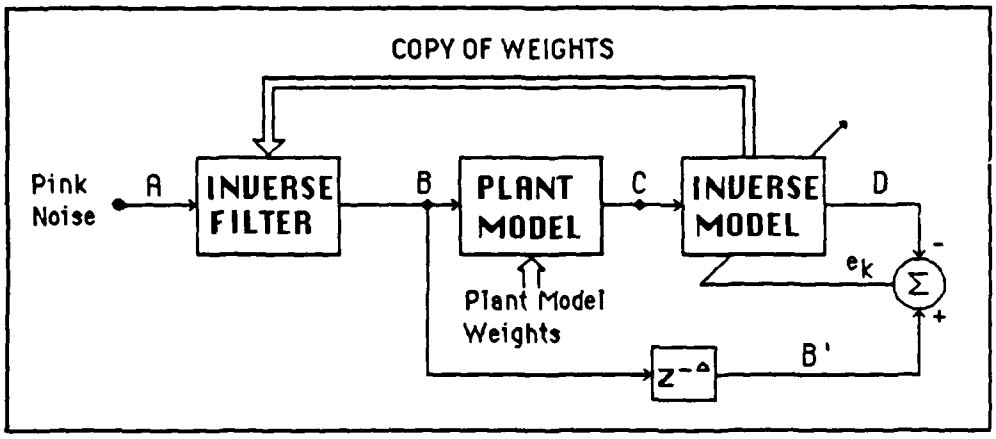


Figure 41. AIMCS Simulation Block Diagram

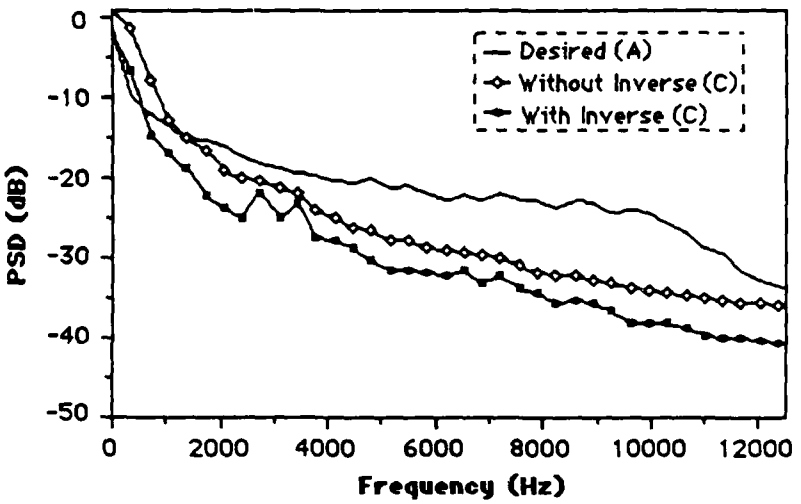


Figure 42. AIMCS PSDs; (A') Desired Signal, (C) Plant Model Output with the Inverse and Without the Inverse Filter

It is the hypothesis of the author that the poor performance is attributed to differences of the input power at the inverse filter and the inverse model inputs. As shown earlier, the inverse model optimum weight vector changes in response to the inverse model's input power. Because of the large low frequency components and the small size of the filter, the optimum inverse model weight vector is constantly changing to track the input power variations due to the large low frequency components. The input power for points A and C as a function of the iteration number is shown in Figure 43. Figure 43 clearly illustrates that the input signal powers would be different at the inverse filter and at the inverse model. Therefore, the inverse filter weight vector, which is the copied inverse model weight vector, would not be the optimum for the inverse filter input power.

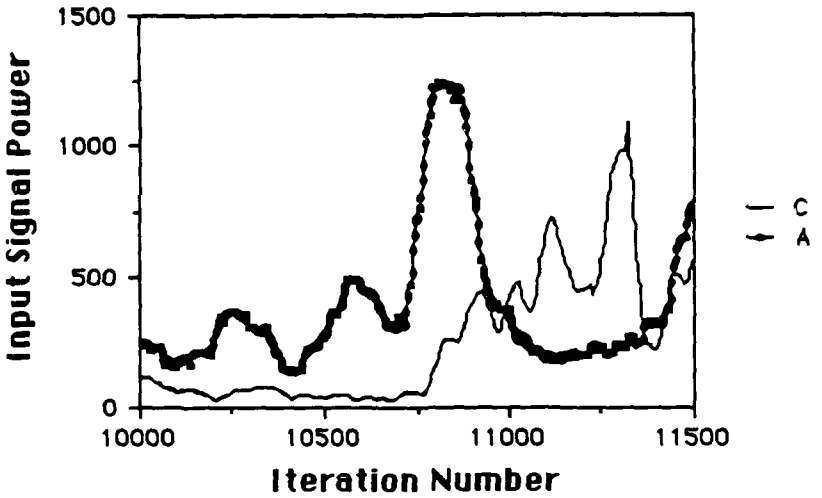
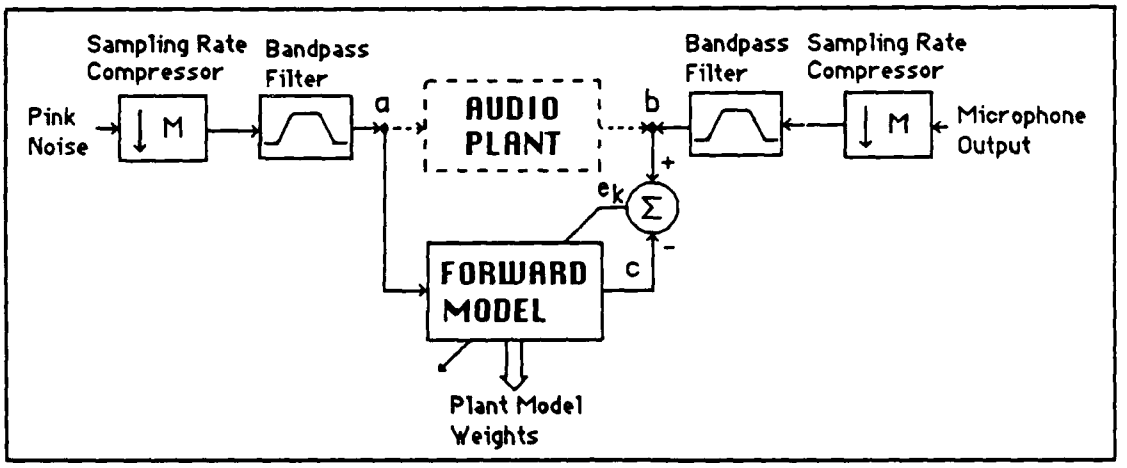


Figure 43. Input Power at A and C

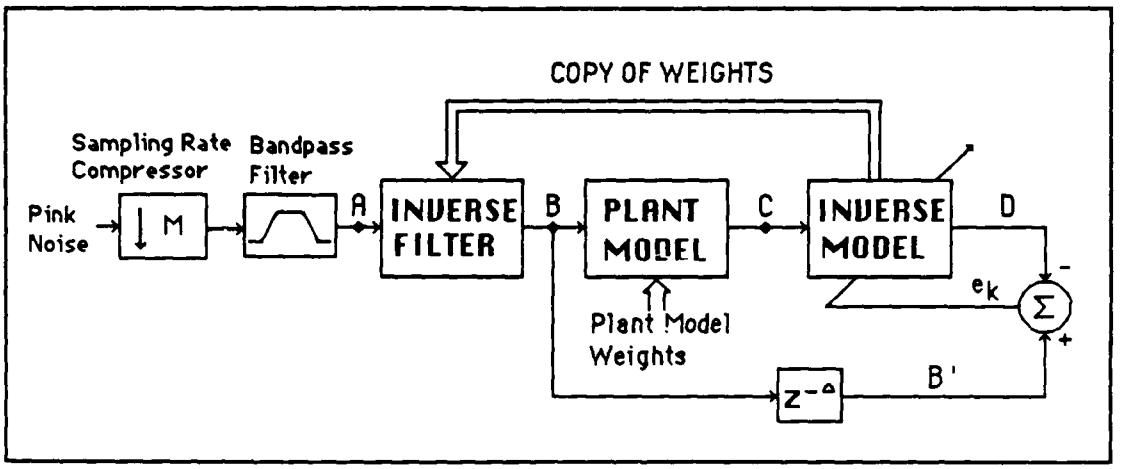
Five Band AIMCS Simulations. A larger AIMCS or a five band AIMCS could reduce the power variations at the inputs to the inverse model and inverse filter. The larger filter is not practical, since the maximum number of taps feasible with the TMS320C30 at the 43956 Hz sampling rate is 151. An alternative solution is a five band AIMCS approach which subdivides the 25 - 12000 Hz audio band into the reverberation chamber's five crossover bands. This five band AIMCS allows the realization of much larger filters for the lower frequency bands.

The five band AIMCS simulations entailed decimation and bandpass filtering of the digitized data for each of the five bands, generation of five plant models using the AFM , and five LMS AIMCS simulations. The simulation process block diagram for one band is shown in Figure 44. No effort was made to recombine the output time sequences from the five simulations. For the five bands, Table

1 lists the sampling rates, the maximum filter size practical for the specified sampling rate, and the decimation factor (M) to reduce the sampling rate.



(a)



(b)

Figure 44. AIMCS Five Band Block Diagrams; (a) AFM, (b) AIMCS

Thirty-one tap FIR bandpass filters were synthesized using the window method with the Bartlett window (Prog 8). The rolloff of the digital bandpass filters were approximately 30 dB per octave, and the 3 dB cutoff points were the lower and upper bandpass limits listed in Table 1.

Table 1. Practical Limits for Five Band AIMCS

Band (Hz)	25-100	100-400	400-1500	1500-5000	5000-12000
Sampling Rate	439.6	1758.2	5494.5	14652.0	43956.0
Number of Taps	15100	3775	1208	453	151
Decimation Factor (M)	100	25	8	3	1

For the initial simulations, the criteria for establishing the number of taps for each of the plant models was the average PSD magnitude difference between points B and C of the AFM simulation. The number of plant model taps was increased until the average PSD magnitude difference between points B and C was below 1.0 dB which was less than the 1.6 dB for the 1000 tap plant model used in the preceding single band AIMCS simulation. Table 2 lists the sizes of each plant model and the corresponding average PSD magnitude difference.

Table 2. Plant Model Specifications

Band (Hz)	25-100	100-400	400-1500	1500-5000	5000-12000
Number of Taps	70	140	140	300	70
Average PSD | b-c |	.8	.6	.9	.6	.8

Table 3 lists the number of AIMCS taps, the average magnitude difference with and without the inverse filter enabled and Figures 45 through 49 show the PSD curves for the five AIMCS simulations. The curves illustrate that the five band AIMCS can effectively inverse filter the plant model since the AIMCS significantly improves the match of the desired and the plant model output when the inverse filter of the AIMCS is enabled.

Table 3. Number of AIMCS Taps

Band (Hz)	25-100	100-400	400-1500	1500-5000	5000-12000
Number of Taps	140	300	300	453	151
Average PSD | A-C | (dB) with Inverse Enabled	1.0	0.6	1.0	1.2	0.7
Average PSD | A-C | (dB) with Inverse Disabled	2.0	2.2	2.2	3.2	5.4

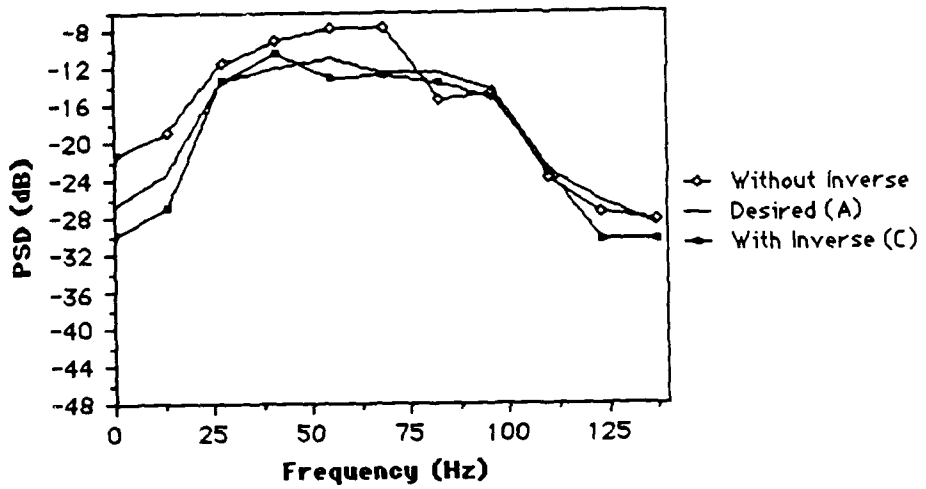


Figure 45. AIMCS PSDs for Band 1

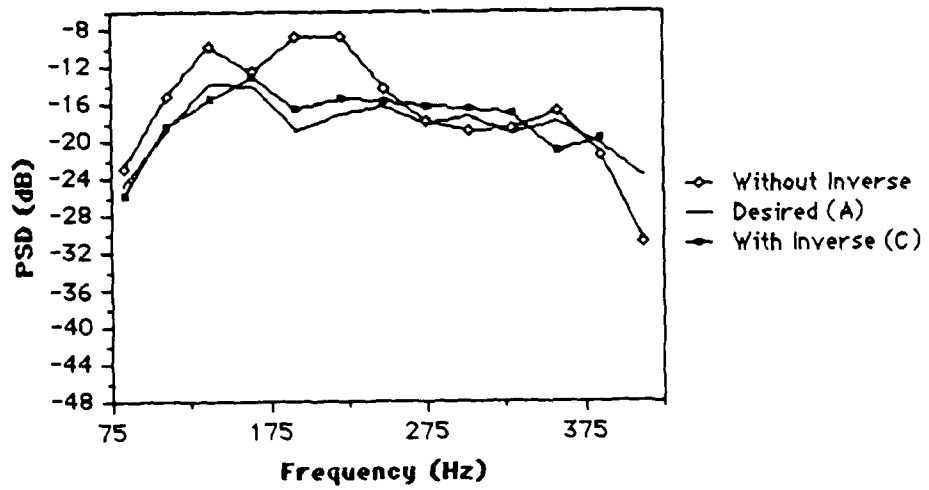


Figure 46. AIMCS PSDs for Band 2

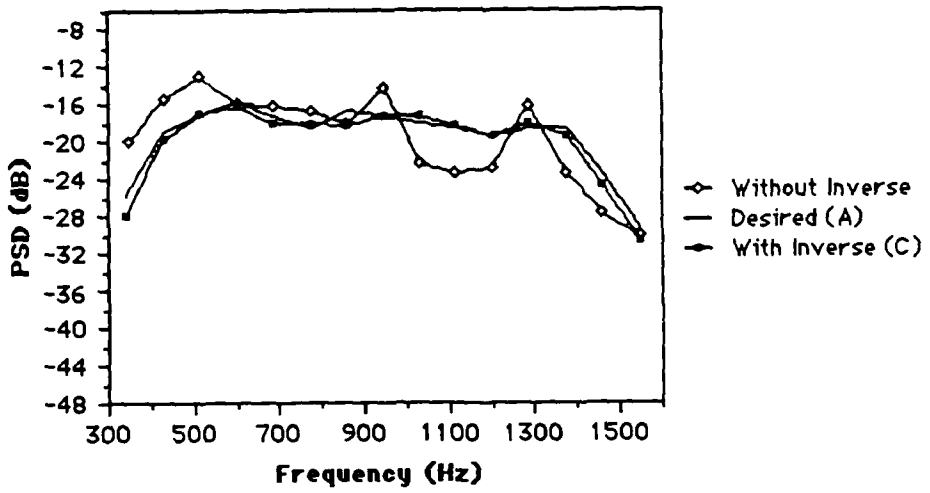


Figure 47. AIMCS PSDs for Band 3

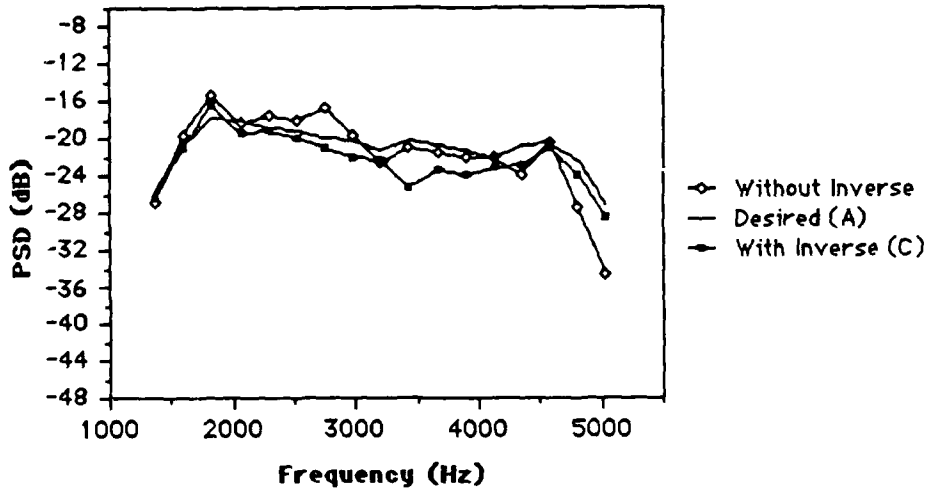


Figure 48. AIMCS PSDs for Band 4

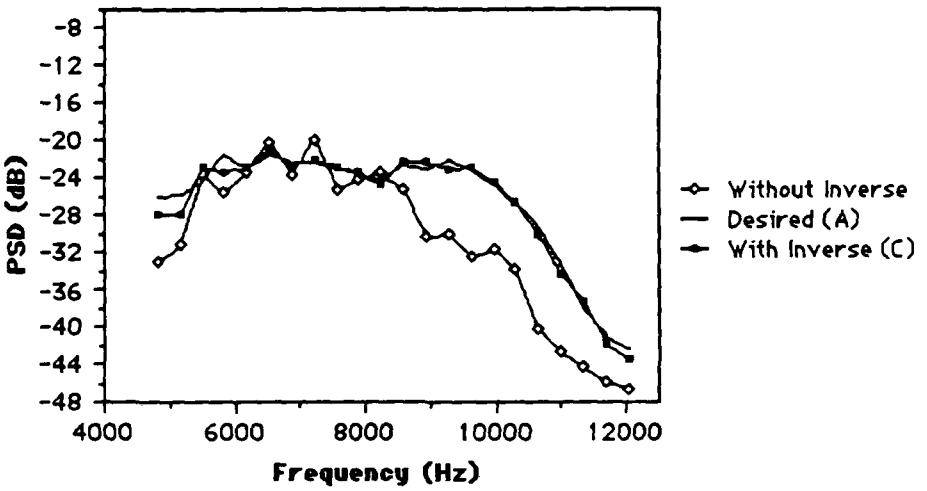


Figure 49. AIMCS PSDs for Band 5

The simulation for the 5000 to 12000 Hz band was repeated with larger plant models. As the size of the plant model was increased, the plant modeling performance improved; however, the AIMCS inverse filtering performance decreased. For plant models of greater than 300 taps, the AIMCS failed to remove any of the distortion effect of the plant model. The degraded performance is due to the inability of the adaptive filters 151 zeros to effectively cancel the 300 or more plant zeros. To exactly cancel the effect of an all zero plant, an all pole inverse is required which places the poles at the location of the plant's zero. The all pole inverse has an infinite impulse response which can be approximated by a long FIR filter. With larger plant models, the 151 tap FIR inverse impulse response

is less effective in approximating the longer inverse impulse response.

Since the actual audio plant transfer function probably consists of both poles and zeros, the inverse simulations would be more representative of a real time implementation if the audio plant was modeled with a autorecursive moving average (ARMA), pole zero, plant model. In addition, the total number of taps to realize the ARMA plant model should be less than the number of taps for the moving average (MA) plant model.

Oliver Muron and Jacques Sikorav demonstrated that a small number of AR coefficients significantly improved their modeling of an audio conference room (8:923). The addition of 20 AR coefficients to a 100 coefficient MA model reduced the observed average squared error by more than 12 dB. While the addition of more than 500 MA coefficients to a 100 coefficient MA model was required to achieve the same error reduction. Their findings suggests that a small ARMA model could model an audio plant as well as a larger MA model. An ARMA audio plant model was beyond the scope of this thesis effort.

Filtered-x Algorithm Simulation. Since the primary inverse filter candidate was the AIMCS, only limited simulations were accomplished with filtered-x algorithm. The results are briefly discussed to prevent a future duplication of effort. The analyses which follow are the author's hypotheses.

The filtered-x simulation was accomplished with the alternate form of the filtered-x algorithm, which is shown in Figure 50 (Prog 9). The simulation with the alternate form of the filtered-x requires

the AFM for the LMS input, and unlike a filtered-x algorithm simulation and the AIMCS simulations does not require a simulated audio plant model. The alternate form converges to the same solution as the standard filtered-x algorithm when its assumed the inverse and the audio plant are commutable (15:187). This assumption is not valid for these simulations since the output of the audio plant and the inverse filter weights are time varying. However, the simulations were still conducted to determine what effect the filtered-x input has on the inverse model performance.

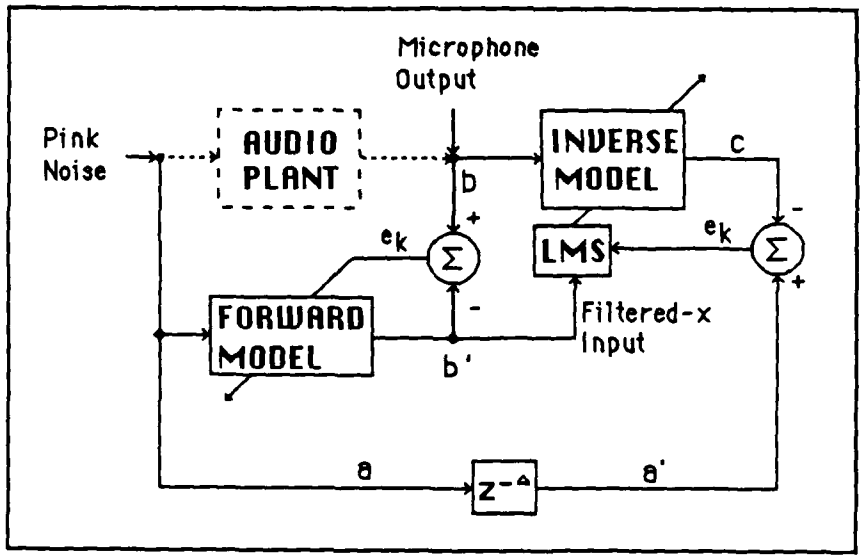


Figure 50. Alternate Form of the Filtered-x Algorithm (15:187)

The AIM and the alternate filtered-x algorithm have almost identical filter structures. The major difference between the AIM and the alternate form of the filtered-x is that the alternate filtered-x algorithm obtains the LMS input from the plant model output while

the AIM obtains the LMS input directly from the plant. The performance of both the AIM and alternate filtered-x algorithm would be identical if the filtered-x algorithm's plant model is an exact copy of the real plant.

The simulation results with a151 tap forward model and inverse model were unsatisfactory since the inverse model did not remove any of the audio plant's distortion effect. In fact, the inverse model contributed to the distortion since the average PSD magnitude difference between points a' and c was larger than the average PSD magnitude difference between points a and b. The average magnitude difference between the PSDs of a' and c was 7.8 dB and between a and b was 5.6 dB. The inefficacy of the alternate filtered-x simulations is attributed to the forward plant model process.

The151 tap forward model ,which approximates the audio plant, only partially correlates the LMS inputs. As was shown for the AIP, the inverse performance degrades when the correlation between the LMS inputs decreases.

Because the LMS inputs are only partially correlated, the maximum convergence constant which allowed stable operation was two orders of magnitude smaller than the convergence constant utilized during the 151 tap AIM simulation. With the smaller u, the alternate filtered-x algorithm will be less capable of tracking the nonstationary $\mathbf{W}^*$. Therefore, the weight misadjustment will be worse than the misadjustment for the AIM simulations. The weight misadjustment could be a major contributor to the poor inverse performance. Widrow and others are investigating the misadjustment of the filtered-x algorithm (15:188).

Simulations were also conducted with a plant model consisting of a pure delay. The pure delay models the propagation delay between the speakers and the microphone, but it does not correct for the phase response of the plant. The transport delay , t_d, between a speaker and the microphone is given by

$$t_d = \frac{d}{v_s} \qquad \text{sec} \qquad (3.8)$$

where d is the distance between the speaker and the microphone in feet and v_s is the velocity of sound in air which is approximately 1000 ft/sec. Delay times ranging from .005 to .012 seconds were simulated to account for the speaker, reference microphone separations of 5 to 12 ft. Again the results were unsatisfactory.

Frequency Domain Adaptive Spectrum Shaper. Since the author was not fully satisfied with the time domain adaptive filter results, a preliminary investigation of a frequency domain adaptive spectrum shaper (FDASS) was accomplished at the end of this thesis effort. The FDASS, which is shown in Figure 51, is immune to the decorrelative effect of the audio plant's phase response since the weight update uses the power spectrum estimates of the LMS inputs. This filter structure is the author's innovative approach to realize a pre-filter structure that can pre-compensate for the spectral distortion of minimum and non-minimum phase plants.

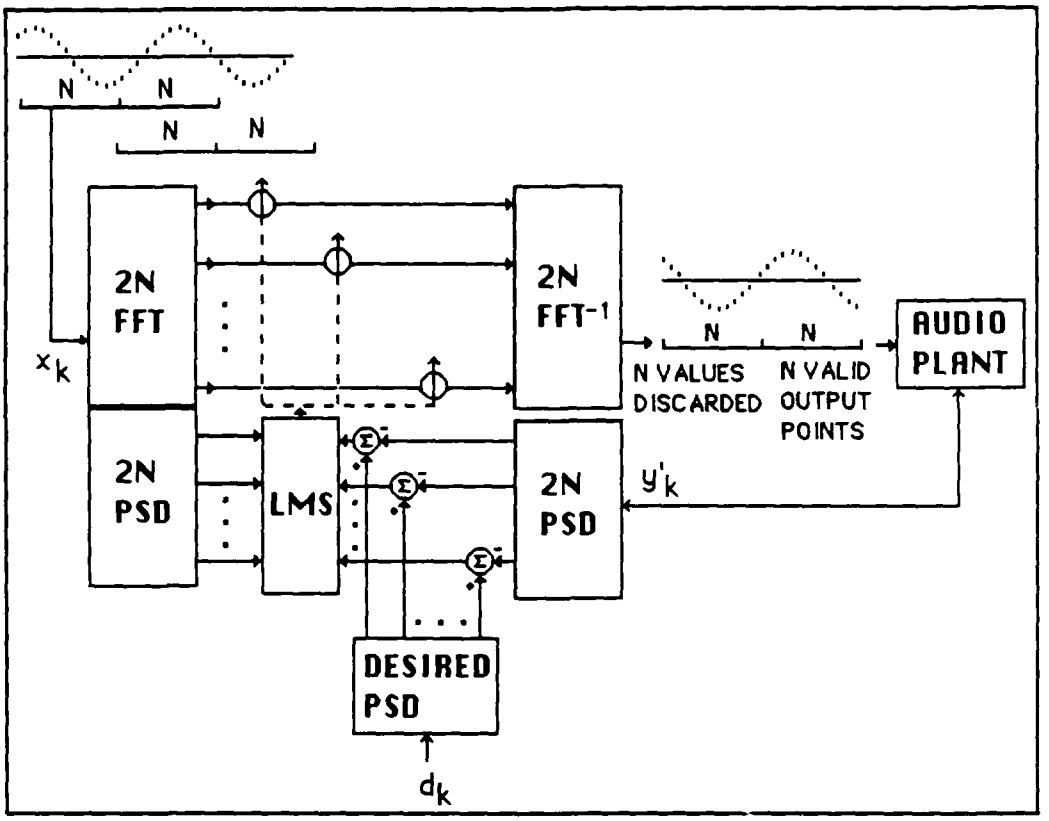


Figure 51. Frequency Domain Adaptive Spectrum Shaper

The basic architecture of the FDASS is based on the complex frequency domain adaptive filter. A brief literature review, which introduces the complex LMS frequency domain algorithm's theory of operation, is presented in Appendix B. Unlike the complex frequency domain adaptive filter, the FDASS's weight update inputs are power spectral estimates and not complex signals. The FDASS uses block processing and the overlap and save fast convolutional technique for calculating the filter output (13:198-201). The input data samples are segmented into 2N points and transformed with an FFT to generate 2N complex frequency samples. The complex

frequency sample in each of the input 2N frequency bins is multiplied by a weight which is controlled by the LMS algorithm. The product is then inverse transformed into 2N real data points. The first N output data points, which are artifacts of the circular convolution are discarded. The last N data points are then read from the data buffer at the appropriate time as a output data point. To compute the next valid N data points, the 2N sample input window is shifted over N points and the entire process is repeated. The process is summarized for the kth block iteration as

$$\mathbf{Y}_k = \mathbf{W}_k \, \mathbf{X}_k \qquad (3.9)$$

where $\mathbf{W}_k$ is the weight vector, $\mathbf{X}_k$ is the input vector

$$\mathbf{X}_k = \mathcal{F}_{2N}[x((k-1)N) \; x((k-1)N+1) \ldots x((k-1)N + 2N-1)] \qquad (3.10)$$

, $\mathbf{Y}_k$ is the ouput vector

$$\mathbf{Y}_k = \mathcal{F}_{2N}[\text{N discarded values } y(kN) \; y(kN+1) \ldots y((k-1)N+2N-1)] \qquad (3.11)$$

,and the symbol $\mathcal{F}$ is the Fourier transform operator.

During the kth block iteration, the LMS algorithm predicts the filter weight vector $\mathbf{W}_{k+1}$ for the (k+1)th block iteration in accordance with

$$\mathbf{W}_{k+1} = \mathbf{W}_k + u \, \mathbf{E}_k \mathcal{P}_{2N}[\mathbf{X}_k] \qquad (3.12)$$

where $\mathcal{P}$ is the power spectral density operator, $\mathbf{u}$ is the convergence constant vector and $\mathbf{E_k}$ is the error spectral estimate. The error spectral estimate vector $\mathbf{E_k}$ is given by

$$\mathbf{E_k} = \mathcal{P}_{2N}[\mathbf{D_k}] - \mathcal{P}_{2N}[\mathbf{Y'_k}] \qquad (3.13)$$

where $\mathcal{P}_{2N}[\mathbf{D_k}]$ is the desired spectrum and $\mathcal{P}_{2N}[\mathbf{Y'_k}]$ is the spectral estimate of the plant model output.

A simulation was accomplished with a 128 weight FDASS and a 100 tap plant model. The results of the simulation are illustrated in Figure 52 (Prog 10). The FDASS effectively precompensated for the spectral distortion of the plant model. The plant model output PSD closely matches the desired PSD when the FDASS is enabled.

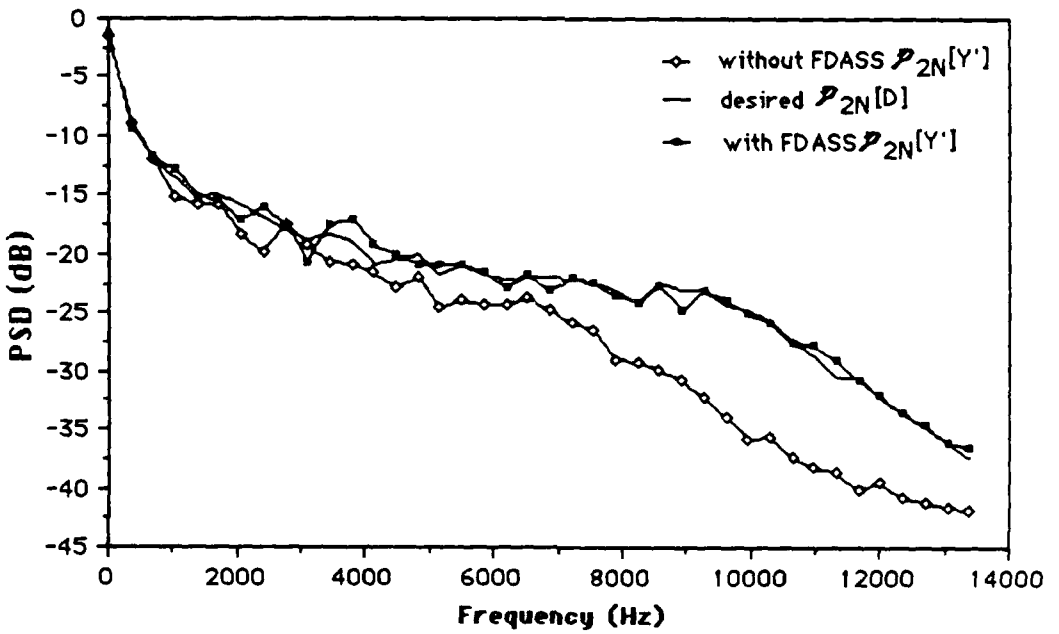


Figure 52. Adaptive Spectrum Shaper PSDs

Summary

This chapter presented the results of experiments and computer simulations to verify the theory developed in Chapter 2 and to analyze the effectiveness of an inverse control system in removing the unwanted distortion effect of an audio plant. The simulations illustrated that the AIP could inverse filter minimum phase plants and the filtered-x algorithm and the AIMCS could inverse filter both minimum and non-minimum phase plants.

An analyses of digitized data from the reverberation chamber demonstrated that the audio plant transfer function distorts the pink noise spectrum and slowly changes with time. The non-stationary behavior of the audio plant lends itself to a hardware implementation of an adaptive system that can automatically track and remove the distortion effect of the audio plant.

The AIMCS and the filtered-x algorithm were analyzed to determine the effectiveness of the two control system in removing the frequency distortion of the audio plant. The only promising time domain simulation was the five band AIMCS which effectively pre-inverse filtered the limited size FIR plant models.

Preliminary results with a frequency domain adaptive spectrum shaper were provided which demonstrated that a frequency domain, adaptive filter could effectively equalize a plant model. Recommendations follow in the next chapter.

IV. Recommendations

This chapter discusses the recommendations for the continuation of this research and for the development of adaptive signal processing design software. The recommendations are directed to AFAMRL and the AFIT Electrical Engineering Department.

Adaptive Inverse Filter

The five band AIMCS was the most promising time domain adaptive filter candidate. Unlike the AIP, the single band AIMCS and the filtered-x, the five band AIMCS was capable of pre-inverse filtering the limited size FIR plant models. Even though it was the best time domain candidate, a frequency domain approach should be thoroughly explored before preceding with a hardware implementation. It is anticipated that the five band AIMCS would be costly to build, since five parallel self contained AIMCSs are required. The following list identifies some of the more expensive components for each of the AIMCS:

a. two TMS320C30 DSPs
b. two A/D
c. one D/A
d. three bandpass filters
e. three sample and holds

In addition to being costly, there is some risk the limited size FIR plant models utilized for the simulations may not accurately model the audio plant. So, the AIMCS five band simulation results may not

be representative of the performance from a real time hardware implementation. To reduce this risk, the simulations could be repeated with an ARMA model that would more accurately model the audio plant.

As an alternative to the five band AIMCS, it is recommended that the FDASS or a complex LMS frequency domain algorithm be considered as a hardware solution. The preliminary FDASS simulation results were very promising. The FDASS was able to effectively compensate for the spectral distortion of a plant model.

For this application, there are two advantages with a frequency domain implementation. First, the frequency domain implementation offers reduced number of computations over the time domain (13:203). The complexity ratio Cr , which is the ratio of frequency domain to time domain real multiplies for an N point impulse response, are shown in Table 4. For a large N, the computational complexity based on number of real multiplies for the

Table 4. Frequency to Time Domain Complexity Ratio

N	Complexity Ratio	
	Complex Frequency Domain (13:203)	FDASS
32	1.20	0.42
64	0.69	0.24
128	0.38	0.14
256	0.21	0.08
512	0.12	0.04
1024	0.06	0.02

frequency domain adaptive filter is much less than an equivalent time domain filter. Thus, with a given DSP throughput, larger adaptive filters can be implemented in the frequency domain.

The second advantage is that the frequency domain adaptive filter provides a means to improve the convergence behavior over a time domain implementation. With a frequency domain adaptive filter, the input spectrum has been divided into spectral bands where each spectral band has its own convergence rate. A convergence constant can be assigned to each spectral band which is dependent on the band's relative input power. By incorporating any a priori information about the power distribution, "the convergence modes of the adaptive filter can be compressed to a more reasonable range, thereby improving the convergence behavior (13:205)". Thus, it can be expected that the convergence behavior of the frequency domain implementation will not be as dependent on the input power variations due to the large low frequency components of the pink noise.

Adaptive Signal Processing Subroutines

In order to reduce the future development time of adaptive signal processing algorithms, an adaptive signal processing software subroutine package should be developed for a PC and/or mainframe. The adaptive signal processing software subroutines would allow a user to efficiently assemble, simulate, and verify adaptive filter

algorithms by calling subroutines from a main program. The following list identifies some suggested subroutines:

a. Time and frequency domain adaptive filters with selectable number of taps.

b. Flexible I/O to import and export digitized data files.

c. Signal analyses to include FFTs, PSDs, autocorrelation and crosscorrelations.

d. High resolution plotting.

e. Lowpass, highpass, and bandpass filters.

f. Signal generator for simulation white noise, sine waves, and linear combinations of signals.

V. Conclusion

This chapter discusses the conclusions culminating from this thesis research. The objective of this thesis was to explore the feasibility of using a time domain adaptive filter to remove the frequency distortion of an audio plant. The approach to accomplish this objective consisted of a review of current literature to find pertinent articles, the development of adaptive inverse filter theory, and the simulation and analysis of the applicable adaptive inverse filter algorithms. In addition, a preliminary investigation of a frequency domain implementation was conducted.

An exhaustive review of the current literature identified the AIMCS and the filtered-x algorithms as prime time domain adaptive filter candidates. Also, the time domain AIP, which was observed by Widrow and Stearns to converge to an irrelevant solution or to be unstable, was also investigated since a preliminary simulation using a three tap minimum phase plant was successful.

Chapter II developed the adaptive inverse filter theory applicable to this thesis. The theory focused on the AIP, filtered-x algorithm, and the AIMCS. The equations for the AIP's and filtered-x algorithm's optimum weight vector and performance function were derived since a complete derivation could not be found in the current literature. A theoretical analysis of a two tap AIP in series with a two tap plant demonstrated that the AIP weight update equation converges to a relevant inverse solution when the plant has minimum phase and an irrelevant inverse solution when the plant

has non-minimum phase. It was shown that the AIP does not converge to a relevant solution for the non-minimum phase plant because the LMS inputs are decorrelated by the plant's large phase response. The filtered-x algorithm compensates for the decorrelative effect of the plant phase response by pre-filtering the LMS input through a plant model.

Chapter III presented and analyzed the results of experiments and computer simulations to verify the theory developed in Chapter 2, to determine the spectral response of the audio plant, and to test the effectiveness of an inverse control system in removing the unwanted distortion effect of an audio plant. Simulations with a 21 tap adaptive filter and a three tap minimum and non-minimum phase plants verified that the AIP could inverse filter minimum phase plants and the filtered-x algorithm and the AIMCS could inverse filter both minimum and non-minimum phase plants.

A spectral analysis of the digitized data from the reverberation chamber illustrated that the frequency response of the audio plant distorts the input signal spectrum and varies slowly with time. The audio plant's non-stationary spectral behavior supports the requirement for an adaptive system to replace the manual graphics equalizer currently in operation. The AIM, which is the adaptive component of the AIMCS, was thoroughly analyzed before conducting the AIMCS simulations. The analyses of the LMS AIM simulations revealed that a single convergence constant, which is bounded by the maximum input power, would not provide the rate of adaptation required to track the low level high frequency components. components. The NLMS AIM, which increases the overall rate of

adaptation by varying the convergence in inverse proportion to the input power, exhibited improved performance over the entire frequency band.

The NLMS AIMCS simulation results however, were unsatisfactory. The poor performance was attributed to difference of the input power at the inverse filter and the inverse model. The only promising time domain adaptive inverse filter simulation was with the five band LMS AIMCS, which effectively removed the distortion effect of the limited size FIR audio plant models.

A preliminary investigation of a frequency domain adaptive spectrum shaper was accomplished at the end of this thesis effort. The frequency domain adaptive spectrum shaper is an innovative approach based on the complex frequency domain adaptive filter architecture that uses block processing and fast output convolution. Unlike the complex frequency domain adaptive filter, the frequency domain adaptive spectrum shaper updates a non-complex weight using power spectral estimates of the LMS weight inputs. Since the LMS weight inputs are not decorrelated by the phase of the plant, the frequency domain adaptive spectrum shaper can find a relevant inverse spectral solution for both minimum and non-minimum phase plants. The simulation results verify that the frequency domain adaptive spectrum shaper is able to precompensate for the spectral distortion of a plant model.

Chapter IV presented a recommendation to thoroughly explore the frequency domain implementation as an alternative approach before preceding with a five band AIMCS hardware implementation. A frequency domain adaptive filter is more computational efficient,

has better convergence properties than a time domain adaptive filter, and may be less expensive to implement than the five parallel AIMCSs comprising the five band AIMCS. It was also recommended that adaptive signal processing subroutines be developed to facilitate the simulation and testing of adaptive filter algorithms.

Appendix A: Computer Programs

All computer programs are written in FORTRAN F77 and were compiled and ran on AFIT's ICC.

Program 1:

```
      PROGRAM AIPMSE
C THIS PROGRAM CALCULATES THE MSE FOR THE AIP FILTER

C DECLARE THE VARIABLES
      REAL  X(50000),Y(50000),W(256),Y1(50000),TNOR,TOTE
      REAL  E(50000),MSE(50000)
      INTEGER DELAY
      OPEN(UNIT=4,FILE='MSE',STATUS='UNKNOWN')
      OPEN(UNIT=1,FILE='NU',STATUS='UNKNOWN')

C INITIALIZE THE VARIABLES
C SEED IS IX
      IX=1
C S IS THE STANDARD DEVIATION
      S=1.0
C AM IS THE MEAN
      AM=0.0
C NPT IS THE NUMBER OF POINTS
      NPT=30000
C NT IS THE NUMBER OF ADAPTIVE FILTER TAPS
      NT = 21
C U IS THE CONVERGENCE CONSTANT
      U = .00009


C INITIALIZE THE MSE VARIABLE WITH ALL ZEROS
      DO 50 N=1,NPT
            MSE(N)=0.0
50          CONTINUE


C THIS  LOOP CALCULATES THE MSE FOR 100 RUNS
C EACH RUN USES A DIFFERENT SEED IX.
      DO 1 P=1,100

C CALL THE SUBROUTINE TO GENERATE THE WHITE GAUSSIAN NOISE FOR THE
C INPUT SEQUENCE
      CALL GAUS(IX,S,AM,NPT,X)

C INCREMENT THE SEED
      IX=IX-1
```

```
C SET THE INVERSE DELAY TO 1/2 THE NUMBER OF TAPS
      DELAY=NT/2

C SET THE PLANT COEFFICIENTS A1-A3
      A1=1.
      A2=-2.8
      A3=2.32

C INITIALIZE THE WEIGHTS TO ZERO.
      DO 5 N=1,NT
            W(N)=0.0
5     CONTINUE


C LOAD THE PLANT OUTPUT VECTOR WITH ZEROS.

      DO 7 N=1,NT
            Y(N)=0.0
7     CONTINUE

      TOTE=0.0
      TNOR=0.0

C MAIN ADAPTIVE FILTER LOOP
      DO 20 N=NT+1,NPT
      ACCUM=0.0

      DO 30 J=1,NT
30    ACCUM=X(N-(J-1))*W(J)+ACCUM

C Y(N) IS THE AIP FILTER OUTPUT
      Y(N)=ACCUM

C Y1(N) IS THE PLANT OUTPUT
      Y1(N)=A1*(Y(N))+A2*(Y(N-1))+A3*(Y(N-2))

C CALCULATE THE ERROR
      E(N)=X(N-DELAY)-Y1(N)

C CALCULATE THE SQUARED ERROR AND THE NORMALIZING FACTOR
      TOTE=E(N)*E(N)
      TNOR=X(N)*X(N)
      IF (TNOR.EQ. 0.0) THEN
         TOTE=0.0
         TNOR=1.0
      END IF

      MSE(N-NT)=(TOTE)+MSE(N-NT)

C PERFORM THE WEIGHT UPDATE
      DO 40 J=1,NT
40          W(J)=W(J)+2.0*U*E(N)*X(N-20-(J-1))
20    CONTINUE
1     CONTINUE
```

```
      DO 80 N=1,NPT-NT-1,100

C CALCULATE THE MSE
      MSE(N)=MSE(N)/P

C WRITE DATA FILES FOR MSE
            WRITE(4,90)MSE(N)
90          FORMAT(F10.4)
      WRITE(1,92)N
92          FORMAT(I5)
80          CONTINUE

    END


C GAUSSAIN NUMBER GENERATOR SUBROUTINE
C THIS SUBROUTINE WAS COPIED FROM THE "DIGITAL FILTER DESIGN
C HANDBOOK" BY TAYLOR (12:101-102)
      SUBROUTINE GAUS(IX,S,AM,MU,F)
            REAL F(*)
            INTEGER N,MU
            REAL A,YFL,AM,S,AVG,TOT
            TOT=0.0
      DO 100 N=1,MU
            A=0.0
            DO 50 I=1,12

C CALL UNIFORM RANDOM GENRATOR
                  CALL RANDU(IX,IY,YFL,TOT)
                  IX=IY
50          A=A+YFL
            F(N)=(A-6.0)*S+AM
100   CONTINUE
      AVG=TOT/(MU*12)
      RETURN
      END

    SUBROUTINE RANDU(IX,IY,YFL,TOT)

      IY=IX*65539
      IF(IY)5,6,6
5     IY=IY+2147483647+1
6     YFL=IY
      YFL=YFL*4.656612875E-10
      TOT=YFL+TOT

      RETURN

      END
```

Program 2:

```
      PROGRAM AIMCSMSE
C THIS PROGRAM CALCULATES THE MSE FOR THE AIMCS FILTER

C DECLARE THE VARIABLES
        REAL X(10000),Y(10000),W(256),Y1(10000),TNOR,TOTE
        REAL E(10000),MSE(10000),Y2(10000),Z(256)
      INTEGER DELAY
       OPEN(UNIT=2,FILE='NU',STATUS='UNKNOWN')
       OPEN(UNIT=4,FILE='MSE',STATUS='UNKNOWN')

C SEED IS IX
      IX=1
C S IS THE STANDARD DEVIATION
      S=1.0
C AM IS THE MEAN
      AM=0.0
C NPT IS THE NUMBER OF POINTS
      NPT=3000
C NT IS THE NUMBER OF ADAPTIVE FILTER TAPS
      NT = 21
C U IS THE CONVERGENCE CONSTANT
      U = .006

C INITIALIZE THE MSE VARIABLE WITH ALL ZEROS
      DO 50 N=1,NPT
       MSE(N)=0.0
50     CONTINUE

C THIS  LOOP CALCULATES THE MSE FOR 100 RUNS.
C EACH RUN USES A DIFFERENT SEED IX.
      DO 1 P=1 100

C CALL THE SUBROUTINE TO GENERATE THE WHITE GAUSSIAN NOISE FOR THE
C INPUT SEQUENCE
      CALL GAUS(IX,S,AM,NPT,X)

C INCREMENT THE SEED
      IX=IX+1

C SET THE INVERSE DELAY TO 1/2 THE NUMBER OF TAPS
      DELAY=NT/2

C SET THE PLANT COEFFICIENTS A1-A3
      A1=1.
      A2=-2.8
      A3=2.32

C INITIALIZE THE WEIGHTS TO ZERO
      DO 5 N=1,NT
       W(N)=0.0
5     CONTINUE
```

```
C LOAD THE PLANT OUTPUT VECTOR WITH ZEROS

      DO 7 N=1,NT
       Y(N)=0.0
       Y1(N)=0.0
7      X(N)=0.0

      TOTE=0.0
       W(11)=.10

C MAIN ADAPTIVE FILTER LOOP
      DO 20 N=NT+1,NPT
      ACCUM=0.0
      ACCUM2=0.0

C CALCULATE INVERSE FILTER OUTPUT
      DO 30 J=1,NT
30      ACCUM=X(N-(J-1))*W(J)+ACCUM
      Y(N)=ACCUM

C CALCULATE PLANT OUTPUT
          Y1(N)=A1*(Y(N))+A2*(Y(N-1))+A3*(Y(N-2))
        E(N)=X(N-DELAY)-Y1(N)

C CALCULATE INVERSE MODEL OUTPUT
      DO 35 J=1,NT
        ACCUM2=ACCUM2+W(J)*Y1(N-(J-1))
35    CONTINUE
      Y2(N)=ACCUM2

C CALCULATE ERRORS
       E2=Y(N-DELAY)-Y2(N)
      TOTE=E(N)*E(N)
       MSE(N-NT)=(TOTE)+MSE(N-NT)

C WEIGHT UPDATE
       DO 40 J=1,NT
40        W(J)=W(J)+2.0*U*E2*Y1(N-(J-1))

20     CONTINUE
1     CONTINUE

      DO 80 N=1,NPT-NT-1,10
       MSE(N)=MSE(N)/P

       WRITE(4,90)MSE(N)
90     FORMAT(F10.4)
      WRITE(2,95)N
95     FORMAT(I4)
80     CONTINUE
       END
```

PROGRAM 3

```
      PROGRAM FILXMSE
C THIS PROGRAM CALCULATES THE MSE FOR THE FILTERED-X ALGORITHM
C DECLARE THE VARIABLES

      REAL  X(10000),Y(10000),W(256),Y1(10000),TNOR,TOTE
      REAL  E(10000),MSE(4000),FILX(10000)
      INTEGER DELAY
      OPEN(UNIT=4,FILE='MSE',STATUS='UNKNOWN')

C SEED IS IX
      IX=1

C S IS THE STANDARD DEVIATION
      S=1.0
C AM IS THE MEAN
      AM=0.0

C NPT IS THE NUMBER OF POINTS
      NPT=3000

C NT IS THE NUMBER OF ADAPTIVE FILTER TAPS
      NT = 21

C U IS THE CONVERGENCE CONSTANT
      U = .00675

C INITIALIZE THE MSE VARIABLE WITH ALL ZEROS
      DO 50 N=1,NPT
            MSE(N)=0.0
50          CONTINUE

C THIS  LOOP CALCULATES THE MSE FOR 100 RUNS
C EACH RUN USES A DIFFERENT SEED IX.
      DO 1 P=1,100

C CALL THE SUBROUTINE TO GENERATE THE WHITE GAUSSIAN NOISE FOR THE
C INPUT SEQUENCE
            CALL GAUS(IX,S,AM,NPT,X)

C INCREMENT THE SEED
            IX=IX+1

C SET THE INVERSE DELAY TO 1/2 THE NUMBER OF TAPS
            DELAY=NT/2

C SET THE PLANT COEFFICIENTS A1-A3
            A1=1
            A2=-2.8
            A3=2.32

C INITIALIZE THE WEIGHTS TO ZERO
             DO 5 N=1,NT
```

```
                W(N)=0.0
5           CONTINUE

C LOAD THE PLANT OUTPUT VECTOR WITH ZEROS.

            DO 7 N=1,NT
                Y(N)=0.0
                FILX(N)=0.0
7           X(N)=0.0

            TOTE=0.0
            TNOR=0.0

C MAIN ADAPTIVE FILTER LOOP
      DO 20 N=NT+1,NPT

            ACCUM=0.0

C CALCULATE THE INVERSE FILTER OUTPUT 'FILX(N)'
            FILX(N)=A1*X(N)+A2*X(N-1)+A3*X(N-2)

            DO 30 J=1,NT
30              ACCUM=X(N-(J-1))*W(J)+ACCUM
            Y(N)=ACCUM

C CALCULATE THE PLANT OUTPUT
      Y1(N)=A1*(Y(N))+A2*(Y(N-1))+A3*(Y(N-2))

C CALCULATE ERROR
      E(N)=X(N-DELAY)-Y1(N)
      TOTE=E(N)*E(N)
      TNOR=X(N)*X(N)
      IF (TNOR.EQ. 0.0) THEN
         TOTE=0.0
         TNOR=1.0
      END IF

      MSE(N-NT)=(TOTE)+MSE(N-NT)

      DO 40 J=1,NT

40              W(J)=W(J)+2.0*U*E(N)*FILX(N-(J-1))

20        CONTINUE
1     CONTINUE

      DO 80 N=1,NPT-NT-1,10
            MSE(N)=MSE(N)/P
            WRITE(4,90)MSE(N)
90          FORMAT(F10.4)
80          CONTINUE

     END
```

Program 4:

```
      PROGRAM AIPASQ
C THIS PROGRAM CALCULATES THE AVERAGE SQUARED ERROR FOR THE AIP
C  AFTER 4000 ITERATIONS HAVE BEEN COMPLETED.  THE WHITE NOISE
C IS COMPUTER GENERATED AND HAS SELECTABLE MEAN AND STANDARD
C DEVIATION.  THE AVERAGED SQUARED ERROR FOR THE 4001 TO THE
C 40000 ITERATION IS SAVED TO THE FILE 'ERRORS'.  THE PROGRAM IS
C CURRENTLY CONFIGURED FOR A TWO TAP PLANT AND TWO TAP ADAPTIVE
C FILTER.  HOWEVER, IT CAN BE EASILY MODIFIED FOR LARGER PLANTS AND
C FILTERS.

C DECLARE THE VARIABLES
      REAL  X(90000),Y(90000),W(256),Y1(90000)
      REAL  Y2(90000),W2(256),Y3(90000),NOISE(90000)
      REAL  E(90000)
       OPEN(UNIT=9,FILE='ERRORS',STATUS='UNKNOWN')

C INITIALIZE THE VARIALBLES
C IX IS THE SEED
      IX=1

C S IS THE STANDARD DEVIATION
      S=1.0

C AM IS THE MEAN
      AM=0.0

C NPT IS THE NUMBER OF POINTS
      NPT=40000

C NT IS THE NUMBER OF ADAPTIVE FILTER TAPS
      NT=2

C CALL THE SUBROUTINE TO GENERATE THE WHITE GAUSSIAN NOISE FOR THE
C INPUT SEQUENCE

      CALL GAUS(IX,S,AM,NPT,X)

C SET THE INVERSE DELAY
      DELAY=2

C SET THE PLANT FILTER COEFFICIENTS
      A1=1.
      A2=5.0
      A3=0.00

C CALCULATE THE AVERAGED SQUARE ERROR FOR A RANGE OF CONVERGENCE CONSTANTS
C STARTING WITH U=.000007.  IT THEN INCREMENTS THE CONVERGENCE CONSTANT BY
C BY .000001 AND RECALCULATES THE AVERAGE SQUARED ERROR.

      DO 2 U=.000007,.000011000,.000001
```

```
              P=0

C INITIALIZE THE ADAPTIVE FILTER WEIGHTS TO ALL ZEROS.
              DO 5 N=1,NT
                    W(N)=0.0
                    W2(N)=0.0
5             CONTINUE


C LOAD THE PLANT OUTPUT VECTOR WITH ZEROS.
              DO 7 N=1,NT
                    Y(N)=0.0
                    Y2(N)=0.0
                    Y3(N)=0.0
7             X(N)=0.0

       TOTE=0.0

C MAIN FILTER LOOP
       DO 20 N=NT+1,NPT

C INITIALIZE THE FILTER ACCUMULATORS WITH ZEROS
              ACCUM=0.0

C CALCULATE THE INVERSE FILTER OUTPUT
              DO 30 J=1,NT
30            ACCUM=X(N-(J-1))*W(J)+ACCUM
       Y(N)=ACCUM

C CALCULATE THE PLANT OUTPUT
        Y1(N)=A1*Y(N)+A2*Y(N-1)+A3*Y(N-2)

C CALCULATE THE ERROR
       E(N)=X(N-DELAY)-Y1(N)
        ERSQ=E(N)**2.0


        IF (N.GT.4000)THEN
              P=P+1
              TOTE=TOTE+ERSQ
        END IF

C UPDATE WEIGHTS
       DO 40 J=1,NT
40              W(J)=W(J)+2.0*U*E(N)*X(N-(J-1))

20            CONTINUE

100                 AVGE=TOTE/P
              WRITE(9,60)U,AVGE,W(1),W(2)
60                  FORMAT(F10.6,2X,F12.7,2X,F10.4,2X,F10.4)

2      CONTINUE
       END
```

Program 5

```
      PROGRAM AIM
C THIS PROGRAM CALCULATES THE INVERSE MODEL OF A
C REVERBERATION CHAMBER AT AMRL/BBA USING THE ADAPTIVE
C INVERSE MODELING.  THE DIGITIZED DATA FILES ARE WNG AND MIC
C  WNG IS THE DATA FROM THE PINK NOISE GENERATOR AND MIC IS THE
C DATA FROM THE REFERENCE MICROPHONE.
C     NT IS THE NUMBER OF TAPS
C      X(N) IS THE FILTER INPUT-DATA FROM THE WNG (WNG=X(N))
C      D(N) IS THE DATA FROM THE MIC (MIC=D(N))
C     E IS THE ERROR SIGNAL
C     Y(N) IS THE FILTER OUTPUT
C     NPT IS THE NUMBER OF SAMPLE POINTS.

C DECLARE THE VARIALBES
      INTEGER XI(52000),DELAY,COUNT,FCOUNT,DELAY2
      integer di(52000)
      REAL  X(52000),Y(52000),d(52000)
      REAL  W(5000),XMAG(1024),YMAG(1024),DMAG(1024)
       COMPLEX XS(1024)
      REAL  ENERGY(52000),U1(52000),ER(52000),E2(52000)
      REAL  W1(1000),Y2(52000),Y3(52000),y4(52000)

C OPEN DATA FILES
C
      OPEN(UNIT=4,FILE='XDYa',STATUS='UNKNOWN')
      OPEN(UNIT=5,FILE='wng',STATUS='OLD')
      open(unit=6,file='mic',status='old')
      OPEN(UNIT=8,FILE='DSPECT',STATUS='UNKNOWN')
      OPEN(UNIT=7,FILE='XSPECT',STATUS='UNKNOWN')
      OPEN(UNIT=9,FILE='YSPECT',STATUS='UNKNOWN')

C SET THE NUMBER OF SAMPLE POINTS IN THE DATA FILES
C WNG_ AND MIC_.
      NPT=32768
C SET THE NUMBER OF TAPS NT FOR THE ADAPTIVE INVERSE FILTER
      NT=151
C SET THE INVERSE DELAY
      DELAY=75
C SET THE NUMBER OF FFT POINTS FOR THE PSD
      NFFT=256
C SET ALPHA FOR THE NORMALIZED LMS.  SET TO 0.0 IF THE LMS IS GOING TO BE USED
      ALPHA=.30

C SET GAMMA FOR THE NORMALIZED LMS
      GAMMA=100
C SET THE CONVERGENCE CONSTANT IF THE LMS IS BEING USED
      U=0.000
C SET THE STARTING POINT (NSTART) FOR THE \P\S\D
      NSTART=16385
```

```
C SET THE FINISH POINT (NFIN) FOR THE \P\S\D
      NFIN=32768
C SET THE NUMBER OF PSD POINTS TO BE WRITTEN TO THE FILES XDV,
C XSPECT, DSPECT, YSPECT.
      NPSD=70
C NOTE:  NFIN - NSTART + 1 MUST BE A MULTIPLE OF NFFT


C ZERO FILL THE FIRST 2000 DATA SAMPLES.

      DO 10 N=1,2000

         X(N)=0.0
         d(n)=0.0
         Y(N)=0.0
         Y2(N)=0.0
         Y4(N)=0.0
10          CONTINUE


C READ IN DATA FROM THE WNG AND MIC DATA FILES.

C DIGITIZED DATA FILES WERE SAVED AS INTEGER FILES.
      DO 30 N=1,NPT
           READ(5,35)XI(N)
35        FORMAT(I8)
           READ(6,36,END=999)DI(N)
36        FORMAT(I8)

30          CONTINUE

999     CLOSE(5)
            CLOSE(6)


C MULTIPLY DATA BY A CONSTANT AND PREPARE IT FOR PROCESSING. THE DIGITIZED
DATA
C IS CONVERTED TO REAL AND APPENDED TO A 2000 SAMPLE LEADER OF ALL ZEROS.
      DO 40 N=2001,NPT+2000

          X(N)=XI(N-2000)*.0043
          D(N)=DI(N-2000)*.0043
40          CONTINUE

42    CONTINUE

C ZERO THE WEIGHTS.

      DO 70 N=1,NT
           W(N)=0.0
70    CONTINUE

C STAR  MAIN FILTER LOOP.
      DO 80 N=2001+nt,2000+NPT
```

```
C SET FILTER ACCUMULATORS TO ZERO.
      ACCUM=0.0
      ACCUM2=0.0

C CALCULATE THE ADAPTIVE INVERSE MODEL OUPUT AND THE INPUT POWER.
      DO 340 K=1,NT
            ACCUM=ACCUM+D(N-(K-1))**2.0
            ACCUM2=ACCUM2+D(N-(K-1))*W(K)
340   CONTINUE

       Y2(N)=ACCUM2
      energy(n)=accum

C FILTER ENERGY IS USED BY THE NORMALIZED LMS

      IF(ALPHA .EQ. 0.0)THEN
            U1(N)=U
            GO TO 121
      END IF

C CALCULATE THE CONVERGENCE CONSTANTS U1(N) FOR THE NORMALIZED LMS.

      U1(n)=alpha/(gamma+energy(n))

121   CONTINUE

C CALCULATE ERROR BETWEEN THE DESIRED AND THE INVERSE MODELING
C FILTER OUTPUT
      E=x(N-DELAY)-Y2(N)
      ER(N)=E


      ERR=2.0*U1(N)*E

C WEIGHT UPDATE
      DO 120 J=1,NT
            W(J)=W(J)+2.0*err*d(n-(j-1))
 120        CONTINUE


80    CONTINUE
85    continue

C CALCULATE POWER SPECTRUM FOR THE FILTER INPUT D(N), THE
C FILTER OUTPUT Y(N) AND THE DESIRED SIGNAL X(N).  THE PSD IS THE
C AVERAGE PERIODOGRAM ESTIMATOR. RECTANGULAR NON-OVERLAPPING
C WINDOWS ARE UTILIZED.

      N=NFFT

C SAMPLING RATE WAS 22.75 US
      T=22.75E-6
```

```
C FOR DIRECT FFT KODE = 1
        KODE=1
        DF=1./(N*T)
        DO 128 J=1,NFFT
              XMAG(J)=0.0
              DMAG(J)=0.0
128           YMAG(J)=0.0

     FCOUNT=0
     COUNT=0

C CALCULATE PSD OF THE MIC DATA WHICH IS SAVED IN THE ARRAY D(J).
      DO 130 J=2000+NSTART,2000+NFIN

           COUNT=COUNT+1
C CONVERT REAL ARRAY INTO COMPLEX ARRAY
            XS(COUNT)=CMPLX(d(J))

C IF WINDOW IS FULL, RUN FFT.
       IF(COUNT.EQ.NFFT) THEN
          CALL FFT(KODE,N,T,XS)

C CALCULATE THE MAGNITUDE OF THE FFT COMPONENTS AND KEEP RUNNING
C TOTAL FOR EACH FREQUENCY BIN.
          DO 140 K=1,NFFT
                XMAG(K)=CABS(DF*XS(K))+XMAG(K)
140       CONTINUE

C REINITIALIZE COUNTER
          COUNT=0

C FCOUNT COUNTS THE NUMBER OF WINDOWS
          FCOUNT=FCOUNT+1
       END IF
130    CONTINUE
       DO 150 J=1,NPSD

C WRITE THE PSD FOR EACH FREQUENCY BIN INTO THE FILE DSPECT.
             XMAG(J)=10*LOG10(XMAG(J)/FCOUNT)
             WRITE(7,155)XMAG(J)
155          FORMAT(F10.6)
150    CONTINUE

       FCOUNT=0
       COUNT=0

C CALCULATE PSD OF THE PINK NOISE DATA WHICH IS SAVED IN THE ARRAY X(J).
       DO 160 J=NSTART+2000,NFIN+2000
             COUNT=COUNT+1
              XS(COUNT)=CMPLX(x(J-delay))
       IF(COUNT.EQ.NFFT) THEN
             CALL FFT(KODE,N,T,XS)
             DO 170 K=1,NFFT
                   DMAG(K)=CABS(DF*XS(K))+DMAG(K)
```

```
170           CONTINUE
              FCOUNT=FCOUNT+1
              COUNT=0
        END IF
160      CONTINUE
        DO 180 J=1,NPSD

            DMAG(J)=20*LOG10(DMAG(J)/FCOUNT)
            WRITE(8,185)DMAG(J)
185        FORMAT(F10.6)
180        CONTINUE

C CALCULATE PSD OF THE ADAPTIVE FILTER OUTPUT
C WHICH IS SAVED IN THE ARRAY Y2(J).
        FCOUNT=0
        COUNT=0
        DO 190 J=NSTART+2000,NFIN+2000
              COUNT=COUNT+1
              XS(COUNT)=CMPLX(Y2(J))
               IF(COUNT.EQ.NFFT) THEN
                    CALL FFT(KODE,N,T,XS)
                    DO 200 K=1,NFFT
                          YMAG(K)=CABS(DF*XS(K))+YMAG(K)
200                 CONTINUE
                    FCOUNT=FCOUNT+1
                     COUNT=0
              END IF
190     CONTINUE

C CALCULATE THE PSD AVERAGE MAGNITUDE DIFFERENCE BETWEEN XMAG AND DMAG
C AND BETWEEN YMAG AND DMAG AND WRITE THE RESULTS IN TABULAR FORMAT TO
THE
C FILE XDYA.
        DIFF=0.
        DIFF0=0.

          WRITE(4,212)
212        FORMAT('FREQ',5X,'DESIRED',5X,'W/O FIL',5X,'W FIL')
        DO 210 J=1,NPSD
              YMAG(J)=10*LOG10(YMAG(J)/FCOUNT)
              DIFF=DIFF+ABS(DMAG(J)-XMAG(J))
              DIFF0=DIFF0+ABS(DMAG(J)-YMAG(J))
              FREQ=(J-1)/T/NFFT
              WRITE(4,215)FREQ,XMAG(J),DMAG(J),YMAG(J)
215           FORMAT(F7.1,2X,F10.6,2X,F10.6,2X,F10.6)
              WRITE(9,220)YMAG(J)
220           FORMAT(F10.6)
210     CONTINUE
        DIFF=DIFF/NPSD
        DIFF0=DIFF0/NPSD
        WRITE(4,225)
225       FORMAT('AVMAG(X-W/O)',3X,'AVMAG(X-W)')
        WRITE(4,230)DIFF,DIFF0
230       FORMAT(F9.4,2X,F9.4)
```

```
C CALCULATE AVERAGE SQUARED ERROR, AVERAGE INPUT POWER,  AND AVERAGE
C CONVERGENCE CONSTANT FOR THE NLMS AND APPEND TO FILE XDVA.
      ERROR=0.0
      ENER=0.0
      ERROR2=0.0
      UCONST=0.0
      dsq=0.0
      dsq2=0.0

      DO 240 J=2001+nstart,2000+NPT
            ENER=ENER+ENERGY(J)
            ERROR=ERROR+(ER(J))**2.
            UCONST=UCONST+U1(J)
            dsq=dsq+x(j-delay)**2.0
240    CONTINUE

      AVGENE=ENER/(NPT-nt)
      UCONST=UCONST/(NPT-nt)
      ERROR=ERROR/dsq
      WRITE(4,245)
245    FORMAT('AVG ENERGY',7X,'AVG U',7X,'AVG SQ IMOD ER')
      WRITE(4,250)AVGENE,UCONST,ERROR
250      FORMAT(F10.4,2X,F13.8,2X,F10.4,)

     CLOSE(4)
     CLOSE(7)
     CLOSE(8)
     CLOSE(9)

     END

C FFT SUBROUTINE WAS COPIED FROM THE "METHODS OF DISCRETE SIGNAL
C AND SYSTEM ANALYSIS" BY JONG (3:262-265)

     subroutine fft(kode,n,delta,x)
c power-of-2 fft (direct and inverse) algorithm.
c
c kode = 1 for direct fft, -1 for inverse fft.
c n = number of samples, must be a power of 2, otherwise error
c message will be printed
c delta = t (sampling interval in seconds) for direct transform
c df (frequency spacing in hz) for inverse transform
c for strict dft (not an approximation to continous ft), set delta
c to 1 for direct transform and to 1/n for inverse transform.
c x = complex array holding data samples (input before, output after).
c
     complex x(*),w1,x1,cmplx
     integer n
     ir=0
     n1 = n
  5  n2=n1/2
    if(n2*2 .ne. n1) go to 100
     ir = ir + 1
```

```
      n1 = n2
      if(n1 .gt. 1) go to 5
       pn = 6.283185/n
      l=n/2
       ir1=ir-1
      k1=0
      do 30 is=1,ir
15    do 20 i=1,l
       k=k1+1
       kp1=k+1
        am=kbitr(k1/2**ir1,ir)
      if(am .ne. 0.) go to 18
       x1 = x(kp1)
      go to 19
18    arg=am*pn
      c=cos(arg)
      s=-kode*sin(arg)
      w1=cmplx(c,s)
       x1=w1*x(kp1)
19     x(kp1) = x(k)-x1
        x(k)=x(k)+x1
20    k1=k1+1
      k1=k1+1
      if(k1 .lt. n) go to 15
      k1 = 0
       ir1 = ir1-1
30    l=l/2
      do 40 k=1,n
        k1 = kbitr(k-1,ir)+1
      if(k1 .le. k) go to 40
       x1=x(k)
        x(k)=x(k1)
       x(k1) = x1
40    continue
      if(delta .eq. 1.) return
      do 50 k=1,n

           x(k)=x(k)*delta

50     continue

      return
100   write(9,101) n
101   format(i6,' is not a power of 2, fft run aborted')
      return
      end

      function kbitr(k,ir)
      kbitr = 0
      k1 = k
      do 1 i=1,ir
      k2 = k1/2
       kbitr = 2*kbitr + k1 - 2 * k2
1     k1=k2
```

```
      return
      end
      subroutine window(z,nfft)
      real z(*)
      do 10 n=1,128
         z(n)=z(n)*(1-cos(6.2832*(n-63)/nfft))
10    continue
      return
      end
```

Program 6:

```
      program formod
c this program calculates the forward model of a
c reverberation chamber at amrl/bba. The forward model
c weights are saved during the last iteration to the file "weights6".
c The program can operate in a decimation or in a normal mode.
c The decimation mode reduces the initial sampling rate and bandlimits the
c digitized data. The forward models generated in the decimation mode are
c used in the five band AIMCS simualtions. The bandpass filter coefficients
c are stored in the data file "bpfw". The file wng_ contains digitized
c data from the noise generator, and the file mic_ contains the digitized data
c from the microphone.
c     nt is the number of taps
c     x(n) is the filter input-data from the mic (mic=x(n))
c     d(n) is the desired sequence-data from the noise gen (wng=d(n))
c     e is the error signal
c     y(n) is the filter output
c     u is the convergence constant
c     npt is the number of sample points.

c declare the variables
      integer xi(74000),di(74000),delay,count,fcount,l,dec
        real x(37000),d(37000),y(37000),er(37000)
       real w(2000),xmag(1024),ymag(1024),dmag(1024)
        real f1(256),wbpf(501),x1(37000),d1(37000)
      complex xs(1024)

c open data files for read and write operations
      open(unit=1,file='bpfw',status='unknown')
      open(unit=2,file='weights6',status='unknown')
      open(unit=4,file='xdy',status='unknown')
      open(unit=5,file='wng',status='old')
      open(unit=6,file='mic',status='old')
      open(unit=8,file='dspect',status='unknown')
      open(unit=7,file='xspect',status='unknown')
      open(unit=9,file='yspect',status='unknown')

c set the number of sample points in the data files
c wng_ and mic_
      npt=65536
c set the number of taps nt
      nt=70
```

```
c set the number of fft points nfft
      nfft=32
c set the convergence constant u
      u=.0075
c set the number of bpf taps
      nt5=31
c set the decimation factor
      dec=100
c set the starting point (nstart) for the PSD
      nstart=336
c set the finish point (nfin) for the PSD
      nfin=655
c set the number of psd points to be written to the files  xdy,
c xspect, dspect, yspect.
      npsd=10
c nfin - nstart + 1 must be a multiple of nfft

c make the filter causal for a max filter size of 2000.
      do 10 n=1,2000
            y(n)=0.0
            x(n)=0.0
            d(n)=0.0
            x1(n)=0.0
            d1(n)=0.0
10    continue

c read in data from the wng_ and mic_ files
       do 20 n=1,npt
             read(6,25,end=998)di(n)
25           format(i8)
20     continue

998    close(6)

       do 30 n=1,npt
             read(5,35,end=999)xi(n)
35           format(i8)
30     continue

999    close(5)

c the number of data points after decimation
       if(dec .ne. 0) then
              npt=npt/dec

c read in BPF weights
              do 43 n=1,nt5+1
                    read(1,45)wbpf(n)
45                        format(f13.8)
43                  continue

       end if
```

```
c multiply data by a constant and filter data for processing.
      l=1
      do 40 n=2001,npt-2000
      if(dec.eq.0)then
            x(n)=xi(n-2000)*.0043
            d(n)=di(n-2000)*.0043
            go to 40
      end if
          x1(n)=xi(1)*.0043
          d1(n)=di(1)*.0043
        accum=0.0
        accum2=0.0
        do 47 k=1,nt5
            accum=accum+wbpf(k)*x1(n-(k-1))
47          accum2=accum2+wbpf(k)*d1(n-(k-1))

c wbpf(nt5+1) is the bandpass filter scaling factor
          x(n)=accum/wbpf(nt5+1)
          d(n)=accum2/wbpf(nt5+1)
        l=l+dec
40    continue

c zero the weights
       do 70 n=1,nt
       w(n)=0.0
70     continue

c repeat the adaptive filter iteration 10 times to insure the filter has converged
     do 85 p=1,10

c main adaptive filter loop
      do 80 n=2001+nt,2000+npt

c set filter accumulators to zero.
     accum=0.0

c calculate the filter output.
     do 110 j=1,nt
110      accum=x(n-(j-1))*w(j)+accum

     y(n)=accum

c calculate error between the desired and the filter output

      e=d(n)-y(n)
     er(n)=e

c calculate the filter weights for the next iteration.
     err=2.0*u*e
      do 120 j=1,nt
            w(j)=w(j)+err*x(n-(j-1))

c check if it is the last iteration
            if(n .eq. npt+1999.and. p.EQ.10) then
```

```
c write weight vector to file weights.
                        write(2,115)w(j)
115                     format(f10.6)
                end if
120     continue

80      continue
85      continue

c calculate power spectrum for the filter input x(n), the
c filter output y(n) and the desired signal d(n).
        n=nfft

c sampling rate was 22.75 us
        t=22.75e-6*dec

c for direct fft kode = 1
        kode=1

         df=1./(n*t)

c initialize PSD accumulators to zero.
          do 128 j=1,nfft
                xmag(j)=0.0
                dmag(j)=0.0
128       ymag(j)=0.0

        fcount=0
        count=0

c calculate the psd for the pink noise data
         do 130 j=2000+nstart,2000+nfin
            count=count+1
              xs(count)=cmplx(x(j))
           if(count.eq.nfft) then
            call fft(kode,n,t,xs)
            do 140 k=1,nfft
                  xmag(k)=cabs(df*xs(k))+xmag(k)
140         continue
            count=0
            fcount=fcount+1
           end if
130     continue

        do 150 j=1,npsd
             xmag(j)=10*log10(xmag(j)/fcount)
             write(7,155)xmag(j)
155            format(f10.6)
150     continue

        fcount=0
        count=0

c calculate the psd for the mic data
```

```
      do 160 j=nstart+2000,nfin+2000
           count=count+1
            xs(count)=cmplx(d(j))

        if(count.eq.nfft) then
           call fft(kode,n,t,xs)
           do 170 k=1,nfft
                 dmag(k)=cabs(df*xs(k))+dmag(k)
170        continue
           fcount=fcount+1
           count=0
         end if
160     continue

      do 180 j=1,npsd
           dmag(j)=10*log10(dmag(j)/fcount)
           write(8,185)dmag(j)
185        format(f10.6)
180     continue
      fcount=0
      count=0

c calculate the psd for the forward model output
       do 190 j=nstart+2000,nfin+2000
           count=count+1
            xs(count)=cmplx(y(j))
         if(count.eq.nfft) then
           call fft(kode,n,t,xs)
           do 200 k=1,nfft
                 ymag(k)=cabs(df*xs(k))+ymag(k)
200        continue
           fcount=fcount+1
           count=0
         end if
190     continue

c compare psds and write results to file "xdy"
      diff=0.
      diffo=0.
           write(4,212)
212        format('freq',5x,'input',7x,'desired',5x,'output')
       do 210 j=1,npsd
           ymag(j)=10*log10(ymag(j)/fcount)
           diff=diff+abs(dmag(j)-xmag(j))
           diffo=diffo+abs(dmag(j)-ymag(j))
           freq=(j-1)/t/nfft
            write(4,215)freq,xmag(j),dmag(j),ymag(j)
215           format(f7.1,2x,f10.6,2x,f10.6,2x,f10.6)
           write(9,220)ymag(j)
220          format(f10.6)
210     continue
      diff=diff/npsd
      diffo=diffo/npsd
       write(4,225)
```

```
225     format('mag(d-x)',6x,'mag(d-y)')
      write(4,230)diff,diffo
230     format(f9.4,2x,f9.4)

c calculate the average squared error
      error=0.0
      ener=0.0
      dsq=0.0
      do 240 j=2001,2000+npt
            error=error+(er(j)**2.0)
          dsq=dsq+d(j)**2.0
240    continue
      error=error/dsq
      write(4,245)
245     format('avg error')
      write(4,250)error
250     format(f10.4)

      close(7)
      close(8)
      close(9)
      END
```

Program 7:

```
        PROGRAM AIMCS
C THIS PROGRAM CALCULATES THE INVERSE FILTER OF A
C REVERBERATION CHAMBER AT AMRL/BBA USING THE ADAPTIVE
C INVERSE CONTROL MODELING SYSTEM.  THE PLANT MODEL WEIGHTS
C WERE GENERATED BY AN ADAPTIVE FORWARD MODELING FILTER.
C
C     NT IS THE NUMBER OF TAPS
C      X(N) IS THE FILTER INPUT-DATA FROM THE WNG (WNG=X(N))
C     E IS THE ERROR SIGNAL
C     Y(N) IS THE FILTER OUTPUT
C     NPT IS THE NUMBER OF SAMPLE POINTS.

C DECLARE VARIALBLES
      INTEGER XI(74000),DELAY,COUNT,FCOUNT,L,DELAY2
        REAL X(37000),Y(37000),X1(37000)
        REAL W(2000),XMAG(1024),YMAG(1024),DMAG(1024)
      COMPLEX XS(1024)
       REAL ER(37000),E2(37000)
         REAL W1(1000),Y2(37000),Y3(37000),Y4(37000),WBPF(31)
       OPEN(UNIT=1,FILE='BPFW',STATUS='OLD')
       OPEN(UNIT=2,FILE='WEIGHTS6',STATUS='UNKNOWN')
       OPEN(UNIT=4,FILE='XDY',STATUS='UNKNOWN')
      OPEN(UNIT=5,FILE='WNG',STATUS='OLD')
      OPEN(UNIT=8,FILE='DSPECT',STATUS='UNKNOWN')
      OPEN(UNIT=7,FILE='XSPECT',STATUS='UNKNOWN')
      OPEN(UNIT=9,FILE='YSPECT',STATUS='UNKNOWN')
C SET THE NUMBER OF SAMPLE POINTS IN THE DATA FILES
C WNG_ AND MIC_.
```

```
      NPT=32768
C SET THE NUMBER OF TAPS NT FOR THE ADAPTIVE INVERSE FILTER
      NT=453
C SET THE NUMBER OF PLANT TAPS
      NT2=300
C SET THE NUMBER OF BPF TAPS.
      NT5=31
C SET THE DECIMATION FACTOR
       DEC=100
C SET THE INVERSE DELAY
      DELAY=NT/2
C SET THE NUMBER OF FFT POINTS NFFT
      NFFT=64
C SET THE CONVERGENCE CONSTANT U
      U=.00170
C SET THE STARTING POINT (NSTART) FOR THE \P\S\D
      NSTART=336
C SET THE FINISH POINT (NFIN) FOR THE \P\S\D
      NFIN=655
C SET THE NUMBER OF PSD POINTS TO BE WRITTEN TO THE FILES  XDV,
C XSPECT, DSPECT, YSPECT.
      NPSD=10
C NFIN - NSTART + 1 MUST BE A MULTIPLE OF NFFT
C MAKE THE FILTER CAUSAL FOR A MAX FILTER SIZE OF 2000.

      DO 10 N=1,2000

           X(N)=0.0
           Y(N)=0.0
           Y2(N)=0.0
           Y4(N)=0.0
           X1(N)=0.0
10     CONTINUE

C READ IN DATA FROM THE WNG_ FILE
      DO 30 N=1,NPT
            READ(5,35,END=999)X1(N)
35          FORMAT(I8)

30     CONTINUE

999    CLOSE(5)

C CALCULATE THE NUMBER OF DATA POINTS AFTER DECIMATION
       IF(DEC .NE. 0)THEN
             L=1
             NPT=NPT/DEC

C READ BPF WEIGHTS
       DO 43 N=1,NT5+1
             READ(1,45)WBPF(N)
45                 FORMAT(F13.8)
43           CONTINUE
       END IF
```

```
C MULTIPLY DATA BY A CONSTANT AND FILTER WITH BPF
      DO 40 N=2001,NPT+2000
       IF(DEC .EQ. 0) THEN
             X(N)=XI(N-2000)*.0043
             GO TO 40
       END IF
           X1(N)=XI(L)*.0043
          L=L+DEC
          ACCUM=0.0
          DO 47 K=1,31
47            ACCUM=ACCUM+WBPF(K)*X1(N-(K-1))

C WBPF(NT5+1) IS THE BANDPASS FILTER SCALING FACTOR
            X(N)=(ACCUM)/WBPF(NT5+1)
40    CONTINUE

C READ IN WEIGHTS FOR THE PLANT SIMULATOR
       DO 310 J=1,NT2
             READ(2,300,END=315)W1(J)
300                FORMAT(F10.6)
310    CONTINUE
315    CONTINUE
C ZERO THE WEIGHTS.

       DO 70 N=1,NT
             W(N)=0.0
70           CONTINUE
       W(1)=1.0

C REPEAT THE ADAPTIVE FILTER ITERATION 2 TIMES TO INSURE THE FILTER HAS
CONVERGED
       DO 85 P=1,2

C START MAIN ADAPTIVE FILTER LOOP
       DO 80 N=2001+NT2+NT,2000+NPT

C SET FILTER ACCUMULATORS TO ZERO.

500      ACCUM=0.0
     ACCUM2=0.0
     ACCUM3=0.0
     ACCUM4=0.0

C CALCULATE THE INVERSE FILTER OUTPUT
       DO 110 J=1,NT
110                ACCUM=X(N-(J-1))*W(J)+ACCUM
       Y(N)=ACCUM

C CALCULATE PLANT SIMULATOR OUTPUT Y2(N)
       DO 340 K=1,NT2
             ACCUM2=ACCUM2+Y(N-(K-1))*W1(K)
340     CONTINUE
       Y2(N)=ACCUM2
```

```
C CALCULATE THE PLANT INVERSE MODEL OUTPUT Y3(N)
      DO 350 K=1,NT
            ACCUM3=ACCUM3+Y2(N-(K-1))*W(K)
350    CONTINUE
      Y3(N-2000)=ACCUM3

C CALCULATE PLANT OUTPUT WITHOUT INVERSE FILTER
      DO 360 K=1,NT2
360               ACCUM4=ACCUM4+X(N-(K-1))*W1(K)
      Y4(N)=ACCUM4

C CALCULATE ERROR BETWEEN THE DESIRED AND THE INVERSE MODELING
C FILTER OUTPUT
      E=Y(N-DELAY)-Y3(N-2000)
     ER(N)=E

C CALCULATE ERROR BETWEEN THE DESIRED AND THE INVERSE FILTER OUTPUT
      E2(N)=X(N-DELAY)-Y2(N)
     ERR=2.0*U*E

C  WEIGHTS UPDATE
      DO 120 J=1,NT
            W(J)=W(J)+ERR*Y2(N-(J-1))
120          CONTINUE

80    CONTINUE
85    CONTINUE

C CALCULATE POWER SPECTRUM FOR THE FILTER INPUT X(N), THE
C PLANT  OUTPUT Y(N), AND THE PLANT WITHOUT THE INVERSE D(N).
     N=NFFT

C INITIAL SAMPLING RATE WAS 43956.0 HZ
     T=22.75E-6*DEC

C FOR DIRECT FFT KODE = 1
     KODE=1

     DF=1./(N*T)

C INITIALIZE PSD ACCUMULATORS TO ZERO.
      DO 128 J=1,NFFT
            XMAG(J)=0.0
            DMAG(J)=0.0
128               YMAG(J)=0.0

    FCOUNT=0
    COUNT=0

C CALCULATE THE PSD FOR THE PINK NOISE DATA
     DO 130 J=2000+NSTART,2000+NFIN
        COUNT=COUNT+1
          XS(COUNT)=CMPLX(X(J))
```

```
      IF(COUNT.EQ.NFFT) THEN
         CALL FFT(KODE,N,T,XS)
         DO 140 K=1,NFFT
               XMAG(K)=CABS(DF*XS(K))+XMAG(K)
140      CONTINUE
         COUNT=0
         FCOUNT=FCOUNT+1
      END IF
130   CONTINUE

    DO 150 J=1,NPSD
         XMAG(J)=10*LOG10(XMAG(J)/FCOUNT)
         WRITE(7,155)XMAG(J)
155       FORMAT(F10.6)
150   CONTINUE

    FCOUNT=0
    COUNT=0

C CALCULATE THE PSD FOR THE PLANT OUTPUT WITH THE INVERSE FILTER
C BYPASSED
     DO 160 J=NSTART+2000,NFIN+2000
        COUNT=COUNT+1
          XS(COUNT)=CMPLX(Y4(J))

      IF(COUNT.EQ.NFFT) THEN
         CALL FFT(KODE,N,T,XS)
         DO 170 K=1,NFFT
               DMAG(K)=CABS(DF*XS(K))+DMAG(K)
170      CONTINUE
         FCOUNT=FCOUNT+1
         COUNT=0
      END IF
160   CONTINUE

    DO 180 J=1,NPSD
         DMAG(J)=10*LOG10(DMAG(J)/FCOUNT)
         WRITE(8,185)DMAG(J)
185      FORMAT(F10.6)
180   CONTINUE

    FCOUNT=0
    COUNT=0
C CALCULATE THE PSD FOR THE PLANT OUTPUT WITH THE INVERSE ENABLED.
     DO 190 J=NSTART+2000,NFIN+2000
        COUNT=COUNT+1
          XS(COUNT)=CMPLX(Y2(J))
      IF(COUNT.EQ.NFFT) THEN
         CALL FFT(KODE,N,T,XS)
         DO 200 K=1,NFFT
               YMAG(K)=CABS(DF*XS(K))+YMAG(K)
200      CONTINUE
         FCOUNT=FCOUNT+1
         COUNT=0
```

```
      END IF
190   CONTINUE

C COMPARE PSDS AND WRITE RESULTS TO FILE "XDY"
      DIFF=0.
      DIFFO=0.

            WRITE(4,212)
212         FORMAT('FREQ',5X,'DESIRED',5X,'W/O FIL',5X,'W FIL')
      DO 210 J=1,NPSD
          YMAG(J)=10*LOG10(YMAG(J)/FCOUNT)
          DIFF=DIFF+ABS(DMAG(J)-XMAG(J))
          DIFFO=DIFFO+ABS(XMAG(J)-YMAG(J))
         FREQ=(J-1)/T/NFFT
           WRITE(4,215)FREQ,XMAG(J),DMAG(J),YMAG(J)
215          FORMAT(F7.1,2X,F10.6,2X,F10.6,2X,F10.6)
          WRITE(9,220)YMAG(J)
220         FORMAT(F10.6)
210   CONTINUE

      DIFF=DIFF/NPSD
      DIFFO=DIFFO/NPSD
       WRITE(4,225)
225     FORMAT('AVMAG(X-W/O)',3X,'AVMAG(X-W)')
      WRITE(4,230)DIFF,DIFFO
230     FORMAT(F9.4,2X,F9.4)

C CALCULATE THE AVERAGE SQUARED ERROR FOR THE INVERSE MODEL
C AND THE INVERSE FILTER.
      ERROR=0.0
      ERROR2=0.0
      DSQ=0.0
      DSQ2=0.0
       DO 240 J=2001,2000+NPT
        ERROR=ERROR+(ER(J))**2.
        ERROR2=ERROR2+(E2(J))**2.
        DSQ=DSQ+Y(J-DELAY)**2.0
        DSQ2=DSQ2+X(J-DELAY2)**2.0
240   CONTINUE
      AVGENE=ENER/NPT
      UCONST=UCONST/NPT
      ERROR=ERROR/DSQ
      ERROR2=ERROR2/DSQ2
       WRITE(4,245)
245   FORMAT('AVG SQ INV ER',2X,'AVG SQ MOD ER')

       WRITE(4,250)ERROR2,ERROR
250     FORMAT(F10.4,2X,F10.4)
      CLOSE(4)
      CLOSE(7)
      CLOSE(8)
      CLOSE(9)

      END
```

Program 8:

```
      program BPF
c this program generates FIR filter coefficients of a digital bandpass filter
c by using the window synthesis technique. the filter coefficients are
c saved to the file 'bpfw' and the frequency response is saved to the file
c 'data'. the first column in 'data' is the frequency and the second column
c is the magnitude.

c declare variables
      real fcoef(1000),freq(256),phase(256)

c open data files
      open(unit=1,file='bpfw',status='unknown')
      open(unit=2,file='data',status='unknown')

c set the sampling frequency in Hz
      fs=5494.5

c set the number of taps. make sure the number of taps is odd.
      nt=31

c set the upper cutoff in Hz
      fh=1500.0

c set the lower cutoff in Hz
      fl= 400.0

      fc=(fh-fl)/2.0
      wc=fc*2.0*3.14159265

      nt2=nt/2

      do 10 n=-nt2,nt2
        arg=wc/fs*n+1e-39

c use a triangular window
        win=1-abs((2*n)/(nt-1))

        wo=(fc+fl)*2.0*3.14159265/fs
        fcoef(k)=win*(sin(arg))/arg*cos(wo*n)

        write(1,20)fcoef(k)
20              format(f13.8)
          k=k+1
10    continue

c calculate scaling factor
        foef(nt+1)=fs/2.0/(fh-fl)
        write(1,22)fcoef(nt+1)
22        format(f13.8)
```

```
      ntps=nt
c calculate frequency response
      call FRERES(ntps,fcoef,freq,phase,m)
      do 40 k=0,40
      fre=k*fs/80.
       write(2,30)fre,freq(k+1)
30     format(f10.4,2x,f10.6)
40    continue
      end


          SUBROUTINE FRERES(NTPS,Z,FREQ,PHASE,m)
C THIS SUBROUTINE CALCULATES THE FREQUENCY AND PHASE RESPONSE OF
C AN FIR FILTER.  THE FIR FILTER COEFFICIENT VARIABLES ARE Z(*).
C THE FREQUENCY RESPONSE IS STORED IN THE VARIABLE FREQ(*), AND
C THE PHASE RESPONSE IS STORED IN THE VARIABLE PHASE(*).

           REAL Z(1000),REAL,IMAG,FREQ(256),PHASE(256)
          TOTPHASE=0.0
          TOTCORR=0.0

          SETFLAG=0.
          DO 10 N=0,79

C THE NORMALIZED FREQUENCY RANGE FROM 0 TO 8 IS DIVIDED INTO
C 21 INTERVALS.

          REAL=0.0
          IMAG=0.0
          DO 20 I=0,NTPS-1

            REAL=REAL+Z(I+1)*(COS(.0785*N*I))
            IMAG=IMAG-Z(I+1)*(SIN(.0785*N*I))

20        CONTINUE

           FREQ(N)=SQRT(REAL*REAL+IMAG*IMAG)
           IF(REAL.EQ.0.0)REAL=1.0E-99

           PHASE(N)=ATAN(IMAG/REAL)
           IF(N.EQ.0.AND.REAL.LT.0.) SETFLAG=1.
           IF(N.EQ.0) GO TO 40
           IF(SETFLAG.eq.0.)THEN
         IF(IMAG.GT.0.0.AND.REAL.LT.0.0)PHASE(N)=3.1416+PHASE(N)
         IF(IMAG.LT.0.0.AND.REAL.LT.0.0)PHASE(N)=-3.1416+PHASE(N)
        IF(IMAG.LT.0.0.AND.REAL.GT.0.0)PHASE(N)=PHASE(N)
         END IF
         IF(SETFLAG.EQ.1)THEN
          IF(IMAG.GT.0.0.AND.REAL.LT.0.0)PHASE(N)=PHASE(N)
          IF(IMAG.GT.0.0.AND.REAL.GT.0.0)PHASE(N)=PHASE(N)-3.1416
          IF(IMAG.LT.0.0.AND.REAL.LT.0.0)PHASE(N)=PHASE(N)
          IF(IMAG.LT.0.0.AND.REAL.GT.0.0)PHASE(N)=3.1416+PHASE(N)
```

```
        END IF
40      TOTPHASE=TOTPHASE+ABS(PHASE(N))
        TOTCORR=TOTCORR+ABS(COS(PHASE(N)))

10    continue

      RETURN
      END
```

Program 9:

```
      PROGRAM FILX
C THIS PROGRAM SIMULATES THE ALTERNATE FORM OF THE FILTERED-X
C ALGORITHM TO CALCULATE THE INVERSE MODEL OF THE REVERBERATION
C CHAMBER AT AMRL/BBA.
C     NT IS THE NUMBER OF TAPS
C      X(N) IS THE DESIRED SIGNAL (WNG=D(N))
C      D(N) IS THE INVERSE MODEL INPUT SIGNAL (MIC=X(N))
C     NPT IS THE NUMBER OF SAMPLE POINTS.

C DECLARE VARIALBLES
      INTEGER XI(52000),DELAY,COUNT,FCOUNT,DELAY2
      INTEGER DI(52000)
        REAL X(52000),Y(52000),D(52000),X1(57000)
       REAL W(5000),XMAG(1024),YMAG(1024),DMAG(1024)
      COMPLEX XS(1024)
       REAL ER(52000),E2(52000),X2(52000)
        REAL W1(1000),Y2(52000),Y3(52000),Y4(52000),D1(57000)

C OPEN DATA FILES
       OPEN(UNIT=4,FILE='XDY',STATUS='UNKNOWN')
      OPEN(UNIT=5,FILE='WNG',STATUS='OLD')
      OPEN(UNIT=6,FILE='MIC',STATUS='OLD')
      OPEN(UNIT=8,FILE='DSPECT',STATUS='UNKNOWN')
      OPEN(UNIT=7,FILE='XSPECT',STATUS='UNKNOWN')
      OPEN(UNIT=9,FILE='YSPECT',STATUS='UNKNOWN')

C SET THE NUMBER OF SAMPLE POINTS IN THE DATA FILES
C WNG_ AND MIC_.
     NPT=32768
C SET THE NUMBER OF TAPS NT FOR THE ADAPTIVE INVERSE FILTER
     NT=151
C SET THE NUMBER OF PLANT MODEL TAPS
     NT2=50
C SET THE DELAY TO 1/2 THE NUMBER OF TAPS
     DELAY=NT/2
C SET THE NUMBER OF FFT POINTS NFFT
     NFFT=128
C SET THE CONVERGENCE CONSTANT FOR THE ADAPTIVE INVERSE FILTER
     U1=.0010
C SET THE CONVERGENCE CONSTANT FOR THE ADAPTIVE PLANT MODEL
     U2=.025
```

```
C SET THE STARTING POINT (NSTART) FOR THE \P\S\D
      NSTART=24577
C SET THE FINISH POINT (NFIN) FOR THE \P\S\D
      NFIN=32768
C SET THE NUMBER OF PSD POINTS TO BE WRITTEN TO THE FILES  XDY,
C XSPECT, DSPECT, YSPECT.
      NPSD=40
C NFIN - NSTART + 1 MUST BE A MULTIPLE OF NFFT

C MAKE THE FILTER CAUSAL FOR A MAX FILTER SIZE OF 2000.
      DO 10 N=1,2000
           X(N)=0.0
           X2(N)=0.0
           D(N)=0.0
           Y(N)=0.0
           Y2(N)=0.0
           Y4(N)=0.0
10     CONTINUE


C READ IN DATA FROM THE WNG_ FILE
          DO 30 N=1,NPT
              READ(5,35)XI(N)
35            FORMAT(I8)
              READ(6,36,END=999)DI(N)
36            FORMAT(I8)
30              CONTINUE

999    CLOSE(5)
       CLOSE(6)


C MULTIPLY DATA BY A CONSTANT AND PREPARE IT FOR PROCESSING.
        DO 40 N=2001,NPT+2000
            X(N)=XI(N-2000)*.0043
           D(N)=DI(N-2000)*.0043
40     CONTINUE

C ZERO THE WEIGHTS.

       DO 70 N=1,NT
             W(N)=0.0
70      CONTINUE

C START MAIN FILTER LOOP
      DO 80 N=2001+NT+500,2000+NPT

C SET FILTER ACCUMULATORS TO ZERO.

     ACCUM=0.0
     ACCUM2=0.0

C CALCULATE THE PLANT MODEL OUTPUT.
       DO 320 K=1,NT2
```

```
320           ACCUM=ACCUM+X(N-(K-1))*W1(K)
      Y2(N)=ACCUM

C CALCULATE THE INVERSE MODEL OUTPUT.
      DO 340 K=1,NT
            ACCUM2=ACCUM2+D(N-(K-1))*W(K)
340    CONTINUE
       Y(N)=ACCUM2

C CALCULATE THE ERROR FOR THE INVERSE MODEL
      E=X(N-DELAY)-Y(N)
     ER(N)=E

C CALCULATE THE ERROR FOR THE PLANT MODEL
      E2(N)=D(N)-Y2(N)

C CALCULATE THE FILTER WEIGHTS FOR THE NEXT ITERATION.
     ERR=2.0*U1*E
      ERR2=2.0*U2*E2(N)

C      INVERSE MODEL WEIGHT UPDATE
       DO 120 J=1,NT
            W(J)=W(J)+ERR*Y2(N-(J-1))
120    CONTINUE

C      FORWARD PLANT MODEL WEIGHT UPDATE
       DO 135 K=1,NT2
            W1(K)=W1(K)+ERR2*X(N-(K-1))
135    CONTINUE

80           CONTINUE
85           CONTINUE

C CALCULATE POWER SPECTRUM FOR THE INVERSE MODEL INPUT X(N), THE
C INVERSE MODEL OUTPUT Y(N) AND THE DESIRED SIGNAL D(N).

     N=NFFT

C SAMPLING RATE WAS 22.75 US
     T=22.75E-6

C FOR DIRECT FFT KODE = 1
     KODE=1
      DF=1./(N*T)

     DO 128 J=1,NFFT
       XMAG(J)=0.0
       DMAG(J)=0.0
128      YMAG(J)=0.0

     FCOUNT=0
     COUNT=0

C CALCULATE THE PSD FOR THE INVERSE MODEL INPUT DATA
```

```
      DO 130 J=2000+NSTART,2000+NFIN
          COUNT=COUNT+1
            XS(COUNT)=CMPLX(D(J))
       IF(COUNT.EQ.NFFT) THEN
          CALL FFT(KODE,N,T,XS)
          DO 140 K=1,NFFT
                XMAG(K)=CABS(DF*XS(K))+XMAG(K)
140       CONTINUE
          COUNT=0
          FCOUNT=FCOUNT+1
       END IF
130    CONTINUE

      DO 150 J=1,NPSD
           XMAG(J)=10*LOG10(XMAG(J)/FCOUNT)
           WRITE(7,155)XMAG(J)
155          FORMAT(F10.6)
150    CONTINUE

      FCOUNT=0
      COUNT=0

C CALCULATE THE PSD FOR THE DESIRED PINK NOISE SIGNAL
       DO 160 J=NSTART+2000,NFIN+2000
          COUNT=COUNT+1
            XS(COUNT)=CMPLX(X(J-DELAY))

        IF(COUNT.EQ.NFFT) THEN
          CALL FFT(KODE,N,T,XS)
          DO 170 K=1,NFFT
                DMAG(K)=CABS(DF*XS(K))+DMAG(K)
170       CONTINUE
          FCOUNT=FCOUNT+1
          COUNT=0
        END IF
160    CONTINUE

      DO 180 J=1,NPSD
           DMAG(J)=10*LOG10(DMAG(J)/FCOUNT)
           WRITE(8,185)DMAG(J)
185        FORMAT(F10.6)
180    CONTINUE

      FCOUNT=0
      COUNT=0
C CALCULATE THE PSD FOR THE INVERSE MODEL OUTPUT.
       DO 190 J=NSTART+2000,NFIN+2000
          COUNT=COUNT+1
            XS(COUNT)=CMPLX(Y(J))
        IF(COUNT.EQ.NFFT) THEN
          CALL FFT(KODE,N,T,XS)
          DO 200 K=1,NFFT
                YMAG(K)=CABS(DF*XS(K))+YMAG(K)
200       CONTINUE
```

```
          FCOUNT=FCOUNT+1
          COUNT=0
        END IF
190     CONTINUE

C COMPARE PSDS AND WRITE RESULTS TO FILE "XDY"
      DIFF=0.
      DIFFO=0.

          WRITE(4,212)
212           FORMAT('FREQ',5X,'DESIRED',5X,'W/O FIL',5X,'W FIL')
      DO 210 J=1,NPSD
           YMAG(J)=10*LOG10(YMAG(J)/FCOUNT)
           DIFF=DIFF+ABS(DMAG(J)-XMAG(J))
           DIFFO=DIFFO+ABS(DMAG(J)-YMAG(J))
          FREQ=(J-1)*43956./NFFT
            WRITE(4,215)FREQ,XMAG(J),DMAG(J),YMAG(J)
215           FORMAT(F7.1,2X,F10.6,2X,F10.6,2X,F10.6)
          WRITE(9,220)YMAG(J)
220           FORMAT(F10.6)
210     CONTINUE
      DIFF=DIFF/NPSD
      DIFFO=DIFFO/NPSD
       WRITE(4,225)
225     FORMAT('AVMAG(X-W/O)',3X,'AVMAG(X-W)')
       WRITE(4,230)DIFF,DIFFO
230     FORMAT(F9.4,2X,F9.4)

C CALCULATE THE AVERAGE SQUARED ERROR FOR THE INVERSE MODEL
      ERROR=0.0
      ERROR2=0.0
      DSQ=0.0
      DSQ2=0.0
       DO 240 J=2001+NSTART,2000+NPT
        ERROR=ERROR+(ER(J))**2.
        ERROR2=ERROR2+E2(J)**2.
          DSQ=DSQ+X(J-DELAY)**2.0
        DSQ2=DSQ2+D(J)**2.0
240     CONTINUE
              AVGENE=ENER/(NPT-NT)
              UCONST=UCONST/(NPT-NT)
              ERROR=ERROR/DSQ
              ERROR2=ERROR2/DSQ2
        WRITE(4,245)
245           FORMAT('AVG SQ INVM ER',2X,'AVG SQ FMOD ER')

        WRITE(4,250)ERROR,ERROR2
250           FORMAT(F10.4,2X,F10.4)
      CLOSE(4)
      CLOSE(7)
      CLOSE(8)
      CLOSE(9)

      END
```

Program 10

```
      program fdass

c this program simulates the frequency domain adaptive spectrum
c shaper
c declare the variables

      integer xi(37000),di(37000),delay,count,fcount,l,count2
        real x(37000),d(37000),y(37000),y2(37000)
       real xmag(2048),ymag(2048),dmag(2048),w1(1000)
        complex xs(1024),dk(32768),xk(32768),xfft(2048),w(2048)
       complex yk(2048),grad(2048),cmplx,conjg,dfft(2048)
        real energy(37000),er(65000),e(2048),u1(2048)
       real f1(256),u,xavg(1024),eavg(1024)
      integer count3,count4
      complex yfft(2048)

c open the data files
      open(unit=1,file='weights100',status='unknown')
      open(unit=2,file='u',status='unknown')
      open(unit=3,file='dye',status='unknown')
      open(unit=4,file='xdy',status='unknown')
      open(unit=5,file='wng',status='old')
      open(unit=6,file='mic',status='old')
      open(unit=7,file='xspect',status='unknown')
      open(unit=8,file='dspect',status='unknown')
      open(unit=9,file='yspect',status='unknown')

c set the number of sample points in the data files
c wng_ and mic_.
      npt=32768
c set the number of taps nt
      nt=128
c set the number of plant taps
      nt2=100
c set the number of fft points for the PSD
      nfft=128
c set the starting point (nstart) for the PSD
      nstart=16385
c set the finish point (nfin) for the PSD
      nfin=30720
c set the number of psd points to be written to the files  xdy,
c xspect, dspect, yspect.
      npsd=40
c nfin - nstart + 1 must be a multiple of nfft
c make the filter causal for a max filter size of 2000.
c read in data from the wng_ and mic_ files
      do 20 n=1,npt
            read(5,25,end=998)xi(n)
25          format(i8)
20          continue
```

```
998    close(5)

      do 30 n=1,npt
           read(6,35,end=999)di(n)
35          format(i8)
            x(n)=xi(n)*.0043
30     continue
999    close(6)

c multiply data by a constant and prepare it for processing.
       do 40 n=1,npt
             xk(n)=cmplx(x(n),0.0)
            d(n)=di(n)*.0043
            dk(n)=cmplx(d(n),0.0)
40     continue

c read in plant weights
      do 310 j=1,nt2
       read(1,300,end=315)w1(j)
300    format(f10.6)
310    continue
315    continue

c generate converge constant u1(k) for each frequency bin based on the PSD
c of the input signal x.
c alpha is the normalization step size constant which controls the rate of
c of adaptation.
      alpha=15
       do 34 n=1,nt
34      xmag(n)=0.0
c average 20 blocks
      do 36 n=1,20
c load fft buffer
      do 37 k=1,nt
37        xfft(k)=xk(k+(n-1)*1024)
      kode=1
      dt=1.0
      call fft(kode,nt,dt,xfft)
      do 38 k=1,nt
38       xmag(k)=xmag(k)+cabs(xfft(k))
36     continue
      do 39 k=1,nt
        u1(k)=alpha/(xmag(k))**2.0/20.0
       write(2,41)u1(k)
41     format(f13.12)
39     continue

c initialize the weights and the output buffer.
      do 70 n=1,nt
       w(n)=cmplx(1.0,0.0)
       yk(n)=(0.0,0.0)
70     continue
      do 75 k=1,nt2+nt/2
75       y(k)=0.0
```

```
c repeat the filter process 4 times to ensure the filter has converged.
      do 85 pp=1,4

c initialize counters
      do 77 k=1,nt
        eavg(k)=0.0
77      xavg(k)=0.0
      count=nt2+nt/2
      count2=0
      count3=0
      count4=0

c main filter loop
      do 80 n=1+nt/2,npt-nt,nt/2

c block the input data sequence
      do 100 p=1,nt
        xfft(p)=xk(p+n-1)
100    continue
      kode=1
      dt=1.0
      call fft(kode,nt,dt,xfft)

c calculate the filter output
       do 110 L=1,nt
        yk(L)=xfft(L)*w(L)
110    continue

c calculate the filter time sequence output.
      kode=-1
      dt=1.0/nt
      call fft(kode,nt,dt,yk)

c through away the first n/2 data points.
      do 120 k=nt/2+1,nt
      count=count+1
        y(count)=real(yk(k))
120    continue

c calculate the plant output sequence.
      do 390 j=1,nt
           count4=count4+1
           accum=0.0
      do 400 k=1,nt2
400       accum=accum+w1(k)*y(count-nt+j-(k-1))
      y2(count4)=accum

c calculate the plant output sequence fft
       yfft(j)=cmplx(accum,0.0)
390    continue
      kode=1
      dt=1.0
      call fft(kode,nt,dt,yfft)
```

```
c calculate the magnitude of xfft, dfft, and yk.
c don't start unless the  plant input shift register is full.
      if(count4.gt.nt2)then
      do 115 l=1,nt
            xmag(l)=cabs(xfft(l))
            ymag(l)=cabs(yfft(l))
115    continue

c calcuate the error
      count3=count3+1
      do 117 k=1,nt
             e(k)=xmag(k)-ymag(k)
             xavg(k)=xavg(k)+xmag(k)
             eavg(k)=eavg(k)+e(k)
            count2=count2+1
             er(count2)=e(k)
117    continue

c weight update when 10 PSD data blocks have been averaged.
      if(count3.eq.10) then
      do 165 k=1,nt
         w(k)=w(k)+u1(k)*eavg(k)*cmplx(xavg(k),0.0)
165    continue
      do 166 kk=1,nt
        eavg(kk)=0.0
166     xavg(kk)=0.0
      count3=0
     end if
     end if
80     continue
85     continue

c calculate power spectrum for the filter input x(n), the
c filter output y(n) and the desired signal d(n).
      n=nfft
c sampling rate was 22.75 us
       t=22.75e-6
c for direct fft kode = 1
      kode=1
      do 128 j=1,nfft
       xmag(j)=0.0
       dmag(j)=0.0
128     ymag(j)=0.0
      fcount=0
      count=0
      do 130 j=nstart,nfin
            count=count+1
             xs(count)=cmplx(d(j))
      if(count.eq.nfft) then
            call fft(kode,n,t,xs)
            do 140 k=1,nfft
                  xmag(k)=cabs(xs(k))+xmag(k)
140           continue
```

```
          count=0
          fcount=fcount+1
      end if
130     continue
      do 150 j=1,npsd
           xmag(j)=10*log10(xmag(j)/fcount)
           write(7,155)xmag(j)
155          format(f10.6)
150     continue
      fcount=0
      count=0
      do 160 j=nstart,nfin
           count=count+1
            xs(count)=cmplx(x(j))
      if(count.eq.nfft) then
           call fft(kode,n,t,xs)
           do 170 k=1,nfft
                 dmag(k)=cabs(xs(k))+dmag(k)
170          continue
           fcount=fcount+1
           count=0
      end if
160     continue
      do 180 j=1,npsd
           dmag(j)=10*log10(dmag(j)/fcount)
           write(8,185)dmag(j)
185          format(f10.6)
180     continue
      fcount=0
      count=0
      do 190 j=nstart,nfin
           count=count+1
            xs(count)=cmplx(y2(j))
      if(count.eq.nfft) then
           call fft(kode,n,t,xs)
           do 200 k=1,nfft
                 ymag(k)=cabs(xs(k))+ymag(k)
200          continue
           fcount=fcount+1
           count=0
      end if
190     continue
      diff=0.
      diffo=0.
           write(4,212)
212     format('freq',5x,'input',7x,'desired',5x,'output')
      do 210 j=1,npsd
           ymag(j)=10*log10(ymag(j)/fcount)
           diff=diff+abs(dmag(j)-xmag(j))
           diffo=diffo+abs(dmag(j)-ymag(j))
           freq=(j-1)*43956./nfft
            write(4,215)freq,xmag(j),dmag(j),ymag(j)
215          format(f7.1,2x,f10.6,2x,f10.6,2x,f10.6)
           write(9,220)ymag(j)
```

```
220         format(f10.6)
210    continue
     diff=diff/npsd
     diffo=diffo/npsd
      write(4,225)
225      format('mag(d-x)',6x,'mag(d-y)')
     write(4,230)diff,diffo
230     format(f9.4,2x,f9.4)
      close(7)
     close(8)
     close(9)
```

Appendix B: Frequency Domain Adaptive Filter Literature Review

This brief literature review summarizes several frequency domain adaptive filter articles. The following notation is used throughout the summary: boldface letters denote vectors or matrices and capitals denote frequency domain variables.

Mauro Dentino and others introduced the first complex LMS algorithm that adaptively filters in the frequency domain (2:1658). They showed that frequency domain complex LMS adaptive filter implementations offer reduced number of computations over the time domain when the number of weights exceed 16. It's expected the number of weights for the thesis solution to far exceed 16 weights since the reverberation period of the reverberation chamber is approximately 400 milliseconds (4). Their frequency domain LMS algorithm block diagram is illustrated in Figure 1. The input signal x_k and the desired signal d_k are accumulated in separate N-point buffer memories to form N-point data blocks. The data blocks are transformed to N complex numbers with an N-point FFT. The complex number in each of the input N frequency bins is multiplied by an independent complex weight which is controlled by the complex LMS algorithm. During the mth block iteration, the complex LMS algorithm predicts the complex filter weight vector $\mathbf{W}_{m+1}$ for the (m+1)th block iteration in accordance with

$$\mathbf{W}_{m+1} = \mathbf{W}_m + 2\,u\,\mathbf{X}^{\dagger}_m\,\mathbf{E}_m \qquad (1)$$

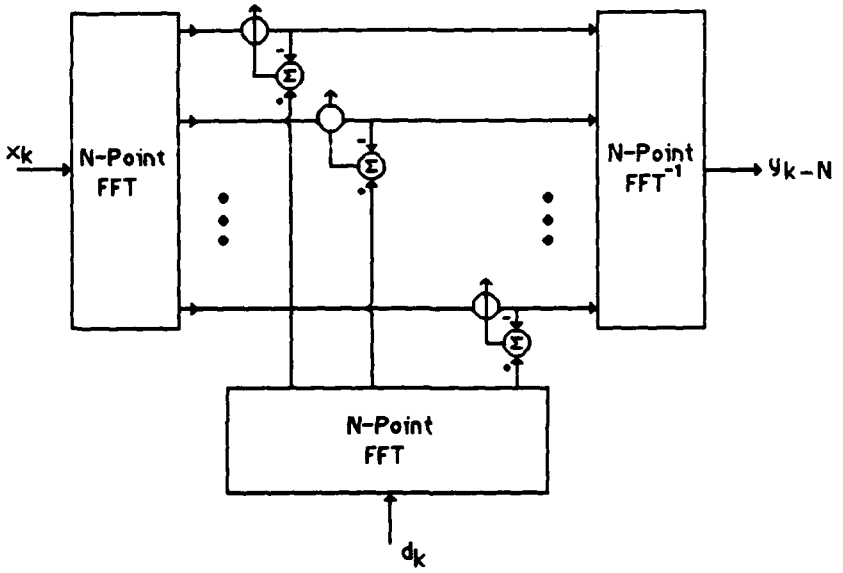


Figure 53. LMS Adaptive Filtering in the Frequency Domain (3:1659)

where u is a non-complex gain constant, $\mathbf{E}_m$ is the error vector, and the character " † " denotes transpose complex conjugate (2:1658). The nth frequency bin complex weight for the m + 1 iteration block is predicted during the mth iteration in accordance with

$$W_n(m+1) = W_n(m) + 2u\, E_n(m)\, X^*_n(m) \qquad (2)$$

where E_n is given by

$$E_n = D_n - Y_n \qquad (3)$$

where D_n is nth desired complex signal and Y_n is the output of the nth complex weight (2:1658). The N complex weighted outputs are transformed with an inverse FFT to an N sample output. The mth

block of the output time sequence is delayed by N number of samples.

Treichler and others presented an intuitive approach for developing frequency domain adaptive filters (13:197). They explained the block processing of the signal data which includes the overlap and save convolution technique. The overlap and save technique circumvents the circular convolution problem due to the periodic nature of the discrete Fourier transform. The technique uses 2N input samples instead of N input samples to compute N output values.

Mohammad Asharif and and others introduced the Frequency Bin Adaptive Filter (FBAF) for noise cancellation in a chamber with multi-reflection impulse response (1:2219-2222). According to the authors, the FBAF performs better than the frequency domain adaptive filter for cancelling noise in a chamber with a long delay impulse response. The long delay impulse response is due to surface multi-reflections of the speaker environment which is a similar to the acoustic environment in the reverberation chamber for this thesis. Each frequency bin of the FBAF is processed by an independent FIR adaptive filter whose weights are controlled by the extended complex LMS algorithm. This linear prediction of past history for each frequency bin makes it possible to compensate for long impulse delays with a shorter window length. The extended complex LMS algorithm for the nth frequency bin and the kth tap delay line is expressed as (1:2220)

$$W_{nk}(m+1) = W_{nk}(m) + 2\,u\,X_{nk}(m)\,E_{nk}(m) \qquad (4)$$

Computer simulations indicate that the FBAF converged faster than the frequency domain adaptive filter. In addition, an FBAF with an eight weight filter and a sample block that was one half as long as the frequency domain adaptive filter outperformed the frequency domain adaptive filter.

Bibliography

1. Asharif, Mohammad R. and others. "Frequency Domain Noise Canceller: Frequency Bin Adaptive Filtering (FBAF)," IEEE Proceedings International Conference on Acoustics, Speech, and Signal Processing, 4: 2219-2222 (1986).

2. Dentino, Mauro and others. "Adaptive Filtering in the Frequency Domain," Proceedings of the IEEE, 66: 1658-1659 (December 1978).

3. Jong, M. T. Methods of Discrete Signal and System Analysis. New York: McGraw-Hill Book Company, 1982.

4. Kofman, Wlodek and Andre Silvent. "Adaptive Estimator of a Filter and Its Inverse," IEEE Transaction on Communications, 33: 1281-1283 (December 1985).

5. Kuc, Roman. Introduction of Digital Signal Processing. New York: McGraw-Hill Inc, 1988.

6. Lim, Jae S. and Alan V. Oppenhiem. Advanced Topics in Signal Processing. New Jersey: Prentice Hall, 1988.

7. McKinley, Richard, Biomedical Engineer. Personal interview. Biological Acoustic Branch of the Armstrong Aerospace Medical Research Laboratory, Wright Paterson AFB OH., 28 April 1988.

8. Murion, Olivier and Jacques Sikorav. "Modelling of Reverberators and Audioconference Rooms," IEEE Proceedings International Conference on Acoustics, Speech, and Signal Processing, 2: 921-924 (1986).

9. Neely, Steven T. and Jont B. Allen. "Invertibility of a room impulse response," Journal of the Acoustic Society of America, 66: 165-169 (July 1979).

10. Simar, Ray and others. "A 40 MFLOPS Digtal Signal Processor: The First Supercomputer On A Chip," IEEE Proceedings International Conference on Acoustics, Speech, and Signal Processing, 1: 535-538 (1987).

11. Stremler, Ferrel G. Introduction to Communication Systems. Massachusetts: Addison-Wesley Publishing Company, 1982.

12. Taylor, Fred J. Digital Filter Design Handbook. New York: Marcel Decker Inc., 1983.

13. Treichler, J. R. and others. The Theory and Design of Adaptive Filters. New York: John Wiley and Sons, 1987.

14. Widrow, Bernard and Samuel D. Stearns. Adaptive Signal Processing. New Jersey: Prentice Hall, 1985.

15. Widrow, Bernard and others. "On Adaptive Inverse Control," IEEE Proceedings 15th Asilomar Conference on Circuits, Systems, and Computers, 6: 185-189 (1981).

16. Widrow, Bernard and others. "Adaptive Control by Inverse Modeling," IEEE Proceedings12th Asilomar Conference on Circuits, Systems, and Computers, 4: 90-94 (1979).

Vita

Captain John R. Strasburger [redacted] He graduated from Scranton Preparatory high school in 1974 and attended the University of Scranton form which he received the degree of Bachelor of Science in Bio-Physics. He received his commission in the USAF from the Officer Training School in 1981. He completed a Electrical Engineering degree conversion program at the Air Force Institute of Technology School of Engineering in March 1983. He then served as an avionics systems engineer with the Aeronautical Systems Division's Directorate of Engineering until entering the School of Engineering, Air Force Institute of Technology, in June 1987.

[redacted]

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/88D-49	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology (AU) Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Armstrong Aerospace Medical Research Laboratory 8b. OFFICE SYMBOL (If applicable) AAMRL/BBA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

ANALYSIS AND SIMULATION OF AN AUDIO ADAPTIVE EQUALIZER (UNCLASSIFED)

12. PERSONAL AUTHOR(S)
John R. Strasburger, Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1988 December	147

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Digital Filters; Adaptive Filters; Audio Frequency; Equalization. (JES)
09	01		
25	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Chairman: Robert Williams, Captain, USAF
Associate Professor of Electrical Engineering

J. S. Przemieniecki
12 Jan 1989

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert Williams, Captain, USAF	22b. TELEPHONE (Include Area Code) (513)-255-6027 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 — Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

The purpose of this thesis was to explore the feasibility of replacing a manual audio equalizer with an adaptive filter that adaptively equalizes the spectral distortion of an audio system. The impulse response of an audio system which includes the response of the speaker crossover network, the power amplifiers, speakers, and the acoustic transfer function between the system's speakers and a reference microphone, distorts an audio system's input signal spectrum. The Adaptive Inverse Pre-filter, the Filtered-x algorithm, and the Adaptive Inverse Modeling Control System are investigated which remove the distortion by pre-filtering the audio system's input signal with the audio system's inverse. The audio system examined is the Armstrong Aerospace Medical Research Laboratory's Performance and Communication Research and Technology reverberation chamber located at Wright-Patterson Air Force Base.

The researcher presents two innovative solutions: a multiband Adaptive Inverse Modeling Control System (AIMCS) and a frequency domain adaptive spectrum shaper. The adaptive spectrum shaper uses an improved weight update algorithm developed specifically for this application. Computer simulation results are presented which demonstrate the effectiveness of the multiband AIMCS and the adaptive spectrum shaper in removing the spectral distortion of an audio system model.